AN INVESTIGATION INTO THE APPLICATION OF DYNAMIC PROGRAMMING TO DETERMINE OPTIMAL CONTROLS FOR TRACKING A CRUISE MISSILE MANUEVER BY CMMCA

THESIS

Leonard G. Heavner
Captain, USAF

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AF T/GST/ENS/89M-6

AN INVESTIGATION INTO THE APPLICATION OF DYNAMIC PROGRAMMING TO DETERMINE OPTIMAL CONTROLS FOR TRACKING A CRUISE MISSILE MANUEVER BY CMMCA

THESIS

Leonard G. Heavner
Captain, USAF

AFIT/GST/ENS/89M-6

Approved for public release; distribution unlimited

AFIT/GST/ENS/89M-6

AN INVESTIGATION INTO THE APPLICATION OF DYNAMIC PROGRAMMING TO DETERMINE OPTIMAL CONTROLS FOR TRACKING A CRUISE MISSILE MANUEVER BY CMMCA

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Operations Research

Leonard G. Heavner, B.S.
Captain, USAF

March 1989

Preface

The study was meant to be a preliminary investigation into the application of dynamic programming to determine the optimal controls to track a cruise missile by the Cruise Missile Mission Control Aircraft (CMMCA). As such, many questions are raised by this study about the application of the methodology to this type of tracking problem.

The concept of a penalty function to maintain the cruise missile in a desired position within the radar sweep cone is introduced. This penalty function is by no means the only possible performance criterion. Indeed, further research is recommended into the true desires of a CMMCA crew they track a cruise missile's flight. Improved definitions of the penalty function are suggested by this investigation.

In the development of ideas that went into the accomplishment of this thesis, I am indebted to my faculty advisor, Lt Col T. F. Schuppe, for his immense patience and availability whenever I requested his assistance. I would also like to thank Dr. J. W. Crissis and Lt Col W. P. Baker for their cooperation over the past few months. Finally, I wish to thank my wife, Melody, for her concern and loving wishes during those many nights I was off pounding the computer keyboard.

Leonard G. Heavner

Table Of Contents

List of Figures

Abstract

This thesis investigated the application of dynamic programming to determine the optimal control inputs for tracking a cruise missile manuever by the Cruise Missile Mission Control Aircraft (CMMCA). Conventional dynamic programming and state increment dynamic programming were considered. The objective was to minimize cruise missile deviations from a desired position within the radar sweep cone. Only single turning manuevers of the cruise missile were investigated in this study. Implementation on a personal computer was attempted. Since time defined the stages for the dynamic programming methodology, the effect of different time increments was evaluated.

The study found state increment dynamic programming difficult to implement. Problems were encountered in the processing of block boundaries. Memory limitations of the FORTRAN compiler prohibited the implementation of the dynamic programming technique on the personal computer. The methodology was successfully implemented on a VAX running on the VMS operating system.

(cont)

The model, as formulated, was found to be extremely sensitive to the size of the time increments defining the stages. Widely different controls were computed using three different time increments. A 0.5 minute time increment was determined to provide the best results of the increments investigated. Keywords: Guided missile tracking optimization. Theses. (cdc)

Further research is recommended to better define the penalty function postulated for the CMMCA tracking problem. Additional investigation is also suggested into the applicability of state increment dynamic programming to reduce high-speed memory requirements and possibly allow implementation on a personal computer.

AN INVESTIGATION INTO THE APPLICATION OF DYNAMIC PROGRAMMING TO DETERMINE OPTIMAL CONTROLS FOR TRACKING A CRUISE MISSILE MANUEVER BY CMMCA

I. Introduction

The mission of the Cruise Missile Mission Control Aircraft (CMMCA) is to track cruise missile test flights and to provide airspace surveillance for collision avoidance. Tracking the cruise missile during off-range test flights is a relatively simple process since the cruise missile's flight path is basically straight with some wide, sweeping turns. On-range test flights, however, present a more challenging scenario. The cruise missile must make a large number of turns to remain within the limits of the test range. At the same time, CMMCA attempts to maintain the maneuvering cruise missile within its radar sweep cone. Because of the differences in the flight characteristics between CMMCA and a cruise missile, tracking the missile is a demanding task for a CMMCA aircrew.

A new radar is being placed onboard the CMMCA to improve CMMCA's ability to keep the airspace around the missile clear of conflicting traffic. Unfortunately, the new radar will have a relatively narrow field of view, both in range

and in azimuth. Though not absolutely necessary, it is desirable to keep the cruise missile within the radar sweep cone during missile test flights. This desire has raised questions concerning how to best track the missile and keep it in the radar's cone.

Problem

The problem addressed by this thesis was to determine the optimal control inputs a CMMCA crew should perform in order to track a cruise missile as it manuevers during a test flight manuever.

Scope

The scope for this research effort was limited to a single cruise missile turn. Combination turns were not considered. The controls modeled in this investigation were bank and thrust.

This effort also considered the effect of changes in the size of the time increment used to define the stages of dynamic programming. The sensitivity of the solution technique with a postulated penalty function was explored.

Background

Three basic techniques were considered to investigate this problem. Simulation was the first technique that seemed appropriate for the question at hand. Simulation, unfortunately, cannot necessarily provide optimal solutions. Dynamic programming was next considered. The large number of

variables involved in this tracking problem required a large amount of high-speed memory for this technique. Another technique considered, state increment dynamic programming, appeared to provide a methodology to handle the large scope of the problem as initially stated.

Assumptions and Limitations

The following assumptions will be made in the solution of the CMMCA tracking problem:

1. The model will work in three-dimensions. However, both the CMMCA and the cruise missile maintain a constant altitude during test flights. All turns performed by both vehicles will be level turns in this model.
2. The model will use nominal values for aircraft capabilities and characteristics. As a mission is flown, an aircraft's weight decreases which in turn affects the aircraft's performance. For simplification, the performance of CMMCA is assumed constant througout a manuever.
3. It is assumed that a cost or penalty function exists for allowing the cruise missile to deviate from the desired position inside the radar sweep cone during a test flight. This research effort postulated one possible formulation of this penalty function. The penalty function formulated is one

meant to minimize cruise missile deviations from a desired position within the radar sweep cone.

4. The model assumed a no-wind condition.

Overview

Chapter 2 presents background on possible solution techniques that appeared promising in the solution of the problem statement. Simulation, conventional dynamic programming, and state increment dynamic programming were considered.

Chapter 3 discusses the methodology of dynamic programming. Since conventional and state increment dynamic programming share common characteristics, both are presented in detail.

The results of applying dynamic programming to this tracking problem are presented in Chapter 4. The difficulties encountered with the state increment methodology are discussed as well as the more promising results obtained with the conventional dynamic programming technique.

Last, Chapter 5 discusses some conclusions and recommendations for future applications of dynamic programming to this problem.

II. Background

Three possible approaches were considered to solve the CMMCA tracking problem. Simulation, dynamic programming, and state increment dynamic programming were considered as alternative techniques.

Simulation

One method to attack the cruise missile tracking problem is to employ a continucus simulation. Schuppe has developed a continuous simulation model that attempts to define how well a CMMCA could track the cruise missile (10:1). The problem is workable as a simulation since the modeling of the movement of two vehicles ic easily accomplished using the equations of motion. On the other hand, the development of a guidance algorithm that mathematically describes how an aircraft follows a missile through a manuever is a more difficult problem.

Schuppe's simulation did not attempt to identify an optimal guidance algorithm (10:1). Instead, he attempted to emulate what a CMMCA navigator might command as he viewed the progress of the missile on his radar screen (10:2). Bank angle and airspeed changes for the CMMCA were based on deviations from a desired position behind the missile. The simulation used maximum roll rates and accelerations regardless of the size of the deviation.

From his model, Schuppe identified a requirement for an improved guidance algorithm (10:10). His simulations demonstrated that simply responding to deviations in azimuth and range appears to be inadequate. Schuppe suggests the need for a guidance algorithm which could predict future missile positions and allow CMMCA to respond before the missile alters its flight path (10:10). Nevertheless, he believed the simulation model can be used as a baseline to measure future improvements.

Dynamic Programming

A methodology that is often well suited for control problems is dynamic programming. This methodology can incorporate the future knowledge of the cruise missile position into its solution technique. Since the flight paths of the cruise missile are planned out in advance of the test flight, future positions of the cruise missile are known in advance. Armed with this future knowledge, dynamic programming can make guidance decisions before the missile's flight path changes.

Problems or processes for which dynamic programming can be applied share the following characteristics:

1. A physical system characterized at any time by a small set of parameters, i.e. state variables;
2. At any time, there exists a choice of a number of decisions, i.e. control variables;

3. The effect of a decision is a transformation of the state variables;
4. The past history of the system is of no importance in determining future actions;
5. The purpose of the process is to maximize or minimize some function of the state variables (1:81-82).

The CMMCA tracking problem follows the characteristics of a dynamic programming problem. There exists a set of parameters that define the system at any time. These parameters would be analogous to the parameters used in a simulation of the problem. Generally stated, the position and velocity of the CMMCA are useful parameters that define the state of the system. Position tells where the CMMCA is currently located while velocity tells where it is going. Together, the two parameters completely describe the CMMCA's state.

At any time, a choice of bank angles and thrusts are available to the crew of the CMMCA. The choices made directly impact the position and velocity of the aircraft. The question of how the CMMCA arrived at its current position and velocity behind the missile is of no importance to the future controls applied to keep the missile in the radar sweep cone. The only question is how to manuever so as to proceed in an optimal way from the current state.

Lastly, the purpose of the tracking process can be defined to minimize the deviations of the cruise missile from some desired position within the radar sweep cone. A penalty function that minimizes the deviations from a desired position does not explicitly exist for this tracking problem. Many possible forms for the function might exist and each form may have a different overall effect on the results of a dynamic programming solution. Assuming a penalty function can be postulated which accomplishes the desired minimization, the tracking problem follows the characteristics of a dynamic programming problem.

All the variables in the CMMCA tracking problems are continuous. The position and velocity of CMMCA as well as any control applied will be defined within a finite range, but may take on an infinite number of values with this range. Time is also a continuous variable. In order to apply the dynamic programming computational procedure, there must be a finite number of admissible states and controls.

This requirement can be met by quantizing the variables. Within each variable's allowable range, each is quantized with a uniform increment. Time is also quantized to provide discrete stages at which each control is tested. The computational procedure can then be applied at these quantized values of stage, state, and control variables. A more in-depth discussion of the computational procedure will be given in Chapter 3.

Bellman and Dreyfus demonstrate this quantization of continuous variables to solve an aircraft time to climb problem (2:209-218). The state variables in their problem were defined to be altitude and velocity. These were quantized into 1000 ft increments for altitude and 0.02 Mach increments for velocity. The only control variable chosen was the climb angle of the aircraft. They assumed maximum thrust throughout the climb. The climb angle was quantized to be either zero for a level acceleration or the angle necessary for a constant velocity climb. The objective function was to minimize the time necessary to climb to a predetermined altitude and velocity.

Through the application of the dynamic programming methodology, they were able to predict the minimum time-to-climb to a desired altitude and velocity. The technique also suggested the required climb profile required to achieve the minimum time. The profile produced proved to be more complicated than a human pilot might be able to follow. Consequently, they believed the most useful purpose of the solution may well be to serve as a guide and criterion to establishing practical minimum-time paths (2:218).

One of the difficulties encountered when applying dynamic programming, especially to a continuous problem, is "curse of dimensionality" (1:9) This curse arises out of the need to maintain at least two storage locations for each quantized state of the system: one for the value of the

objective function and one for the optimal control. The number of locations required is then

$$N_h = 2 \prod_{i=1}^{n} N_i \qquad (2.1)$$

where N_i is the number of quantized values of the ith variable and n is the number of state variables (7:34). Thus, if there are 3 state variables and 100 quantization levels in each variable, then the required number of memory locations is 2,000,000. A high dimension problem with just a few quantized values could saturate a computer system.

State Increment Dynamic Programming

Larson conceived state increment dynamic programming as an optimization procedure that has all the desirable properties of the conventional dynamic programming computational procedure, but requires much less high-speed memory (7:39). He accomplishes this reduction by introducing two additional concepts: the block and the time over which a control is applied.

The concept of the block is the primary vehicle by which Larson accomplishes the reduction in required high-speed memory. Blocks are defined by partitioning the (n+1)-dimensional space containing the n state variables plus time into rectangular sub-units (7:46). Each block contains a small number of each quantized state and quantized time. Quantized states along the boundary of a block are considered

to belong to all blocks that share that boundary. This sharing allows for some transition capabilities between blocks.

Each block is processed individually. When all computations are completed in a block, that block is swapped out of high-speed memory into bulk memory, and a new block is processed. Overall, high-speed memory requirements can then be controlled by selectively choosing the number of quantized states contained within the block. The successful processing of the blocks depends on the next concept.

A fundamental difference between state increment dynamic programming and conventional dynamic programming is the method for determining the time interval over which a given control is applied (7:43). In conventional dynamic programming, the next state is always taken to occur in one time increment after the present state. Thus, each control is applied for a fixed amount of time and there is no guarantee that the next state will be near the present state. State increment dynamic programming, however, determines the minimum time interval required for any one of the state variables to change by one increment. Each control might then be applied for different lengths of time, but in no case will a state variable be allowed to change by more than one increment. This requirement ensures the next state lies near the current state.

It is this difference that enables state increment dynamic programming to reduce the high-speed memory requirements from those of conventional dynamic programming (7:46). If the next state is close to the present state, most of the computations can be kept within the block. Only those states that lie along the boundaries of a block will require special computational effort. Controls applied at these boundaries might transition into the adjacent blocks which may not reside in high-speed memory. Larson proposes several techniques to handle such occurrences (7:80). These techniques will be discussed in Chapter 3.

Larson demonstrates an application of state increment dynamic programming in a minimum-fuel trajectory problem for a supersonic transport (6:135-140). This problem is similar to the time-to-climb problem described earlier. State variables are again altitude and velocity with the same increments of 1000 ft and 0.02 Mach respectively. The set of admissible controls for this problem were defined as follows: climb at constant velocity, dive at constant velocity, accelerate at constant altitude, decelerate at constant altitude, and cruise at constant altitude and velocity. Thrust and climb angles for each of these admissible controls were then defined. The objective function was to minimize the fuel used during the hypothetical flight.

Larson defined the blocks for the state increment technique to contain three quantized values for each of the

state variables plus three quantized values of time. The requirement for high-speed memory was only 54 storage locations: one location for the value of the objective function and one location for the optimal control at each of the 27 quantized states in the block. The minimum-fuel profile was generated for a 1000 mile flight in approximately 1 hour of computing time on an IBM 7090 (6:142).

Conclusion

The tracking of a cruise missile during a test flight appears to require future knowledge of the missile's position for CMMCA to adequately track it. This requirement suggested a technique with the characteristics of dynamic programming to estimate a flight profile that minimizes the deviations from a desired position. Due to the high dimensionality of the problem, however, some method to reduce high-speed memory requirements would be helpful. State increment dynamic programming provides the technique which reduces this memory requirement. This research effort explored the use of both conventional and state increment dynamic programming.

III. Methodology

The optimization problems to which dynamic programming have been applied are variously called dynamic optimization problems, variational control problems, or optimization problems for multistage decision processes (7:12). Whatever the name, three essential elements exist for both conventional and state increment dynamic programming problems: the system equations, the performance criterion, and the constraints. Once these elements were defined, a problem statement was made, and the solution process initiated. While the overall process for state increment dynamic programming was essentially the same as the conventional technique, some minor modifications were made to employ the state increment methodology. A method to evaluate the performance of the solution for the three time increments was also defined. Additionally, since the cruise missile's position was required at each time stage for the evaluation of the penalty function, a simulation of the cruise missile's flight was written.

System Equations

The system equations define the relationships between three sets of variables: the stage variable, the state variable, and the control or decision variable.

Stage Variable. The stage variable determines the order that events occur within the system. This variable can

be either continuous or discrete. If it is continuous, such as time, it can be denoted as t and is usually defined over an interval $t_o <= t <= t_f$. On the other hand, if it is discrete it can be defined as the sequence k = 0,1,...,K. When the stage variable is continuous, it is quantized into increments, perhaps denoted as dt when time is involved. The quantized values of t that lie in the range for which t is defined can be indexed by the discrete sequence k = 0,1,...,K where the value of t corresponding to k is given by

$$t = t_o + k\,dt \tag{3.1}$$

and where

$$K\,dt = t_f - t_o \text{ (7:12).} \tag{3.2}$$

This method of quantizing time into discrete elements was employed for the solution of the CMMCA tracking problem. To explore the effect of the time increment size on the solution process, three time increments were used in this research effort: 0.3 minute, 0.5 minute, and 0.7 minute. The number of time intervals, or stages, in each solution process was dependent upon the missile's length of flight.

State Variables. The state variables are a set of variables that completely describe the system. They describe the system in the sense that if their values are known for all k, k = 0,1,2,...,K, then any question about the behavior of the system for this range of k can be answered (7:12)

The choice of a set of state variables for a particular system is not unique. Determination of a suitable set is dependent on the nature of the mathematical model of the system. If n non-redundant state variables can be specified, they can be denoted $x_1, x_2, \ldots, x_n$ and are often written as an n-dimensional vector, X, called the state vector.

For the CMMCA tracking problem, the state vector consisted of four elements. The first two elements were the x and y position of CMMCA on the Cartesian reference frame with the x-axis aligned with east and the y-axis aligned with north. One unit along either axis will represent 1 nautical mile. The third element used was the CMMCA's heading, h, in radians and defined as the angle between the y-axis and CMMCA's velocity vector in a clockwise direction. The last element in the state vector chosen was the CMMCA's true airspeed, v, in nautical miles per minute. Thus, the state vector was written:

$$X(t) = \begin{vmatrix} x(t) \\ y(t) \\ h(t) \\ v(t) \end{vmatrix} \qquad (3.3)$$

In order to apply a dynamic programming solution technique, the state variables must also be quantized as discussed earlier. For this problem, the increments for each state variable were

$$
\begin{aligned}
dx(t) &= 2 \text{ nautical miles} \\
dy(t) &= 2 \text{ nautical miles} \\
dh(t) &= 45 \text{ degrees} \\
dv(t) &= 0.583 \text{ nautical miles/min.}
\end{aligned} \tag{3.4}
$$

The choice of these increments was based on a desire to keep the number of quantized states at a manageable level. For this initial investigation into the application of dynamic programming to the tracking problem, the size of these increments were judged to provide adequate coverage of the state space.

Control Variables. The control variables are those variables in the process that can be modified or controlled. These variables influence the process by affecting the state variables in some prescribed fashion. There are, in general, q control variables denoted $u_1, u_2, \ldots, u_q$. These variables are generally arranged in a q-dimensional vector, U, called the control vector. The control variables used for CMMCA were bank angle, b, and thrust, p. The control vector was written:

$$
U(t) = \begin{vmatrix} b(t) \\ p(t) \end{vmatrix} \tag{3.5}
$$

For this investigation, thrust was not defined as an application of force. Rather, it was thought of as a ratio of the maximum acceleration, or deceleration, available to the CMMCA crew. For example, a thrust of 1.0 was considered the thrust required for a maximum acceleration in airspeed while -1.0 represented a maximum deceleration. Maximum accel-

eration and maximum deceleration are not necessarily equal for the CMMCA in flight. However, this research assumed the maximum acceleration and deceleration were equal.

It was assumed that bank angles and thrust applications were attained instantaneously. As the time increment used to define the stages of dynamic programming is made smaller, the assumption will make the solution less valid. Since the minimum time increment was 0.3 minute (18 seconds) and maximum bank angles and thrust can be attained in a few seconds, this assumption should have little effect on the problem.

The set of admissible controls chosen for this problem were: constant heading and speed, turn right at constant speed, turn left at constant speed, accelerate on a constant heading, and decelerate on a constant heading.

System Equations. The system equations describe how the state variables of stage k+1 are related to the state variables at stage k and the control variables at stage k. If the stage variable is time, the system equation may be written

$$X(t+dt) = X(t) + f[X(t),U(t),t]dt \qquad (3.6)$$

where f is an n-dimensional vector function (4:250).

Given the above state and control vectors for the CMMCA tracking problem, the vector function becomes

$$f[X(t),U(t),t] = \begin{vmatrix} v(t)\sin[h(t)] + g\tan[b(t)]\cos[h(t)]t \\ v(t)\cos[h(t)] - g\tan[b(t)]\sin[h(t)]t \\ g\tan[b(t)] / v(t) \\ \text{constant} \end{vmatrix} \quad (3.7)$$

The derivation of these system equations is provided in Appendix A.

Performance Criterion

The performance criterion evaluates a particular control policy. This criterion can be either a reward function to be maximized or a cost function to be minimized. The performance criterion depends on the value of U(t) in the control sequence, and also on each value of the state vector, X(t). If the criterion is denoted as J, it can be written

$$J = \int_{t_o}^{t_f} L[X(t),U(t),t]\,dt + O[X(t_f),t_f] \quad (3.8)$$

where L and O are scalar functions representing cost per unit time and terminal cost, respectively (4:250). The remainder of this discussion assumes the performance criterion is a penalty function to be minimized.

The performance criterion does not explicitly exist for the CMMCA tracking problem. The approach used for this research effort was to minimize the time-weighted deviations from a desired location ahead of the CMMCA. There are two types of deviations from a desired position within the radar sweep cone. The first deviation is pure range, R, measured

in nautical miles. The second is azimuth, A, measured in radians. To achieve similar units for these two deviations, azimuth deviations were converted to miles by treating the deviation as an arc segment with length (RxA). Thus, the objective function was to minimize the time-weighted mileage from an optimal range, R^*. This function was written as

$$L[X(t),U(t),t] = |R^*-R| + |R^*A|. \qquad (3.9)$$

Absolute values were taken to ensure that only positive mileage deviations were obtained. This same function will also be used as the terminal penalty function.

Constraints

The constraints place restrictions on the values that the state variables and control variables can assume. For this research effort, arbitrary constraints were placed on the range of the x(t) and y(t) variables. The range of the heading variable is defined by the degrees in a circle. Constraints that define the performance of CMMCA come from an interview with Horton (3) and from operational experience. The constraints were stated as follows:

$$\begin{array}{rcccll} 0 & \leq & x(t) & \leq 150 & \text{(nautical miles)} & \\ 0 & \leq & y(t) & \leq 150 & \text{(nautical miles)} & \\ 0 & \leq & h(t) & \leq 360 & \text{(degrees)} & \\ 5.5 & \leq & v(t) & \leq 7.83 & \text{(nm per min)} & \\ -30 & \leq & b(t) & \leq 30 & \text{(degrees)} & \\ -1 & \leq & p(t) & \leq 1 & \text{(\% of min/max)} & (3.10) \end{array}$$

Problem Statement

The optimization problem can now be stated as:

Given

1. A system described by equation 3.6 and 3.7
2. Constraints as stated by equation 3.10, and
3. An initial state

Find:

The control sequence $u(t_o),u(t_1),\ldots,u(t_f)$ such that J in equation 3.8 is minimized while satisfying the constraints.

Solution Process

Bellman's Principal of Optimality. The heart of dynamic programming is Bellman's principle of optimality. This principle states:

> An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision (1:83).

In other words, the minimum cost for state X at stage t is found by choosing the control that minimizes the sum of the cost function to be paid at the present stage and the minimum cost in going to the end from the state at time t+dt which results from applying this control. By applying this principle, the control sequence that minimizes J is obtained.

Computational Process. Computationally, the solution technique begins at the final stage, t_f. The terminal penalty is computed for each quantized state. These penal-

ties represent the penalties incurred by the CMMCA if it were to arrive at these quantized states at the end of the manuever.

After all the terminal penalties are computed, the preceding stage (i.e. next to last stage) is processed. Again, at each quantized state, the penalty is computed for arriving at that state at this stage. Each control is then applied. Any control that results in a violation of a constraint is not allowed. For example, the deceleration control is not allowed if the current state is already at the minimum allowable velocity.

The next state after applying a control for one time increment will likely not be a quantized state. The penalty for arriving at this non-quantized state is interpolated from adjacent quantized states. The sum of the current penalty and the value of the penalty for the next state is then the penalty incurred by applying the control. Thus, the optimal control is the one with the least total penalty.

After all the optimal controls hved been computed for each quantized state, the optimal controls for the current stage are saved in low-speed memory. The next stage is then processed in the same manner. These computations are repeated until the first stage, representing t_o, is processed.

The final step in the solution process is to establish the control sequence that minimizes the penalty function.

First, the set of optimal controls for the stage at t_o is retrieved. If the initial state of the system is a quantized state, the optimal control is found directly by the optimal control for that state. If the initial state is not a quantized value, an interpolation is performed from adjacent quantized states to determine the optimal control. The control is applied and the next state determined. This process was repeated until the final stage is reached.

For the CMMCA tracking problem, linear interpolations were used for both the interpolation of penalties and controls. First, this interpolation procedure was the simplest to implement. The increments used were considered small enough to allow the use of a linear interpolation without large errors. And since the exact nature of the penalties and optimal controls at the quantized states is unknown, a linear interpolation seems advisable as a first-cut approximation for this initial investigation.

State Increment Modifications

The size of the blocks used in the state increment technique must be defined. For the CMMCA tracking problem, the block sizes were defined to contain five quantized states of x(t) and y(t), five quantized states of h(t), three quantized states of v(t), and eight quantized states increments of time. High speed memory requirements for each block was

then 3000 locations for each evaluation of the penalty function and 3000 for each optimal control.

Unlike conventional dynamic programming, the time a control was applied was only as long as it took for one state variable to change by one increment. The next state that resulted after applying the control for the prescribed amount of time will also not likely be a discrete state. Interpolation was used to compute the penalty of the next non-quantized state.

As mentioned in the previous chapter, a strategy was needed for controls that result in a transition to a block not yet computed. The simplest technique Larson suggested is to exclude all controls which result in such a transition during the computation of a block (7:80). He suggested the transition be considered only after both blocks have been computed.

Larson also suggested that a more accurate procedure for processing these transitions would be to extrapolate the minimum penalty function to the not-yet-computed block (7:83). He suggested this extrapolation be accomplished by fitting a low-order polynomial in the state variables to values of the minimum penalty at quantized states on or near the boundary.

For this effort, the first method was chosen to transition between blocks. The primary reason for this choice was the ease of application.

Measuring the Performance of the Three Time Increments

Some criteria is required to measure the effect of the different time increments on the dynamic programming solution is required. Three criteria were chosen for this investigation.

1. Would a CMMCA crew judge the controls as appropriate for tracking the cruise missile. This criteria would be a judgement decision.
2. The profile generated should minimize the average time-weighted deviations from a desired position. This criteria would be an evaluation of how well the penalty function performed using the dynamic programming methodology. Since the number of stages for each time increment is dependent upon the flight time of the missile, an average over time of these deviations was used as a performance measure.
3. The profile should minimize the maximum deviations from the desired position was used. For two time increments with average deviations approximately equal, this investigation looked at the minimum range and maximum azimuth that resulted if the controls generated were followed.

Cruise Missile Position

In order to evaluate a penalty at the quantized states, the cruise missile's position must be known at the discrete times which define the stages in the solution technique. A

small simulation of a missile's flight was written using the equations of motion. The cruise missile's position was required every 0.3, 0.5, or 0.7 minutes to correspond with the stages defined in the above formulation. The missile's position was recorded every 0.1 minutes, however, in order to validate the performance of the optimal controls derived from the application of this technique.

Implementation

This investigation into the CMMCA tracking problem was an exploration of the possibilities of dynamic programming. While the solution technique can be implemented on a main frame computer, a personal computer was also considered as a possibility for the implementation of this solution technique. The personal computer would enable quick evaluations of the controls that might be required to follow the missile. Since a FORTRAN compiler is readily available for personal computers, it was chosen as the language for this application.

Conclusion

The system equations, performance criterion, and constraints were established for the CMMCA tracking problem. Within this framework, the dynamic programming problem statement was then made. The solution technique of conventional dynamic programming was next explored. Though the solution technique of state increment is similar to conven-

tional, two modifications were required. A method of evaluating the sensitivity of dynamic programming to the size of the time increment was defined. A description of how the cruise missile position is known at any time was presented along with an overview of the implementation strategy planned.

IV. Results

A dynamic programming model was successfully implemented to determine the optimal controls required to track a missile by CMMCA. Though it was chosen to provide a means to control the high-speed memory required of dynamic programming, state increment dynamic programming proved to be difficult to implement. The state space as presented in Chapter 3 was reduced to keep the size of the problem within the high-speed memory capabilities of a personal computer. The FORTRAN compiler for the personal computer, unfortunately, did not allow enough computational capacity for even the reduced state space. A successful application of dynamic programming was accomplished on a VAX mainframe. An evaluation of the different time increment size was made from the mainframe runs. Four common characteristics present themselves after using dynamic programming for tracking missile turns.

State Increment Dynamic Programming

State increment dynamic programming was chosen as a solution technique to control the high-speed memory requirements of dynamic programming. Problems were encountered with the technique mainly because of the high dimensionality of the problem as formulated.

The first difficulty encountered with state increment dynamic programming was the process of interpolating a value for the penalty function when the next state was not a

quantized state. The dimensionality of the tracking problem as formulated is five--four state variables plus time. Since a control is applied until at most one state variable changes by one increment, interpolation was required for the remaining three state variables plus time. Performing a linear interpolation in four dimensions required values for the penalty function at 16 quantized points. The relative positions of the required 16 points to one another within the block was dependent upon the current state and the control applied. The changing relative positions made the interpolation process difficult.

The most difficult problem encountered, however, was processing the quantized states along the boundaries of each block. The method of excluding controls which result in transitions into adjacent blocks proved to be unsatisfactory. The accuracy of the penalty incurred along the boundaries was diminished by using this technique. These inaccuracies were then compounded when controls were applied and the next state was close to the block boundaries. The boundary processing technique chosen appears to be limited.

These conclusions were arrived at from some initial applications of state increment dynamic programming. To track a missile in a straight flight path, the optimal controls that were produced call for bank and thrust applications. This did not agree with expectations. The solution

should have showed the CMMCA following a straight flight path behind the missile.

Return to Conventional Dynamic Programming

With the failure of state increment dynamic programming as formulated, an attempt to reduce the high-speed memory requirements was made by reducing the size of the state space. This reduction was primarily necessitated by the high dimensionality of the CMMCA tracking problem. To reduce the memory requirement, the x-y grid was reduced significantly. The increment size remained at two miles for each dimension but the constraints for allowable x(t) and y(t) values were redefined to describe a smaller state space. Those constraints were rewritten as

$$\begin{aligned} 0 &<= x(t) <= 28 \quad \text{(nautical miles)} \\ 0 &<= y(t) <= 28 \quad \text{(nautical miles)}. \end{aligned} \tag{4.1}$$

All other constraints remained the same.

Still, at any discrete time stage, a rather large high-speed memory requirement exists. There are 15 discrete values of x and y, eight discrete values of heading, and five discrete values of airspeed in this reduced problem. Thus, 9000 discrete states exist at every discrete time stage. For every discrete state, a memory location must be allocated for the penalty at the last stage processed, the penalty at the current stage, and the optimal control applied

at the current stage. The high-speed memory requirement increased to 27,000 memory locations.

FORTRAN further requires four bytes of memory to store the data for one memory location. To implement dynamic programming and solve the reduced tracking problem would thus require 108,000 bytes of memory. While personal computers with sufficient high-speed memory are available, the FORTRAN compiler used only allowed 65,536 bytes of memory for variables and constants (9:53). Even this reduced problem became too big to be implemented on the personal computer.

Implementation on the VAX

The FORTRAN code was next loaded onto a VAX running under the VMS operating system. The dynamic programming technique was successfully executed with the increased processing power provided by the VAX. The source listing of the code is provided in Appendix B.

The program is menu driven. The first step is to enter a cruise missile manuever. The missile's speed, starting coordinates, begin turn coordinates, bank angle in the turn, end stopping coordinates are required during this step. From this information, a simulation is executed of the flight of the missile. Its position is recorded every 0.1 minutes.

The next step is to initiate the dynamic programming portion of the program. Once initiated, the program computes the optimal control at each quantized state and saves the

result in low-speed memory. Control is returned to the main menu after all stages have been completed.

The user can now retrieve the optimal controls for any initial state of the CMMCA. The initial position, heading, and airspeed behind the missile can be specified. If the initial state is not a discrete state, the program will interpolate the initial controls required for this initial state. The flight of CMMCA is then simulated. The results of this simulated flight are printed to the screen as well as to the file 'RESULTS.DAT ' The results for three turns of a cruise missile at each of the three time increments under investigation are given in Appendix C.

Evaluation of Different Time Increment Sizes

The dynamic programming model as formulated proved to be extremely sensitive to the size of the time increment used to define the stages. A 90, 180, and 270 degree turn was executed for each of the time increments.

The 0.3 minute time increment was expected to produce better results over the other two time increments. This was not the case in any of the turns. By the first criteria, it would be unlikely that a CMMCA crew would follow the controls from any of the turns. The 90 degree turn produced a bank to the right followed by a bank to the left ending on another right bank. Both the 180 and 270 degree turns resulted in the optimal controls driving the CMMCA into a position almost directly over the missile. These results

would not be considered as appropriate by the crew. On the first criteria alone, the results of the 0.3 minute increments were unsatisfactory. This result is likely due to the proposed formulation of the penalty function used in this investigation.

The 0.5 minute and 0.7 minute increments both provided more realistic results when compared against the first criteria. The second criteria of average deviations from the desired position proved to be a less conclusive measurement than hoped. The following table summarizes the results.

Table I. Average Time Weighted Deviations in Miles

	0.5 Minute	0.7 Minute
90 Degree Turn	4.53	5.29
180 Degree Turn	7.94	7.94
270 Degree Turn	9.12	8.87

No clear best choice could be made based on this criteria.

The last criteria provided the deciding factor. The maximum deviations observed from the desired location were less with a time increment of 0.5 minute than the 0.7 minute increment. For example, in the 270 degree turn, the 0.5 minute interval produced a minimum range of 4.92 and a maximum azimuth of 110 degrees. The 0.7 minute interval, on the other hand, yielded a minimum range of 4.03 and a maximum azimuth of 180 degrees. By these criteria, the 0.5 minute time interval is the best time interval for the CMMCA tracking problem of those tested.

Common Characteristics of Turns

From the turns executed by this investigation, four characteristics of the dynamic programming solution become apparent. First, when the CMMCA is matching the cruise missile speed prior to the missile's turn, the optimal control to apply prior to the turn of the missile is a deceleration. All three of the sample manuevers have this control as the initial response. This result appears to be caused by the closure that occurs as the missile begins its turn.

Second, both bank and thrust controls are called for simultaneously in the dynamic programming solution. The set of allowable controls, however, used to produce these results did not include simultaneous application of bank and thrust. For example, the set of controls considered at each quantized state allowed either a bank or a thrust change, or neither, but not both at the same time. This innnnnnnn nresult is realistic of how a CMMCA aircrew would actually track a cruise missile.

Third, small control inputs may present themselves as the optimal control at any stage. Bank angles of two or three degrees cannot be accurately maintained by a human pilot. These small control inputs may be caused by the interpolation process in the retrieval of optimal controls. These small controls, though suggested as optimal by this

methodology, will not be practical during actual test flights.

Finally, as the CMMCA starts to fall in behind the missile after the completion its turn, small oscillations occur. A small right bank may be called for at one stage followed by a small left bank in the next. Oscillations may be caused by the size of the increments used to quantize the state variables. The increments chosen for this solution are relatively large. A finer resolution might reduce these oscillations.

Conclusions

While it is a technique designed to control high-speed memory requirements, state increment dynamic programming requires accurate transitions across the boundaries to be successfully implemented. Conventional dynamic programming's large requirement for high-speed memory is the limitation for successful implementation on a personal computer using FORTRAN. The methodology as formulated and implemented on a mainframe is also highly sensitive to the size of the time increment used to define the stages. For the 0.5 minute time increment, insight is gained into the optimal controls from the characteristics observed from the cruise missile turns demonstrated.

V. Conclusion and Recommendations

Conclusions

Tracking of a cruise missile by CMMCA has proven to be an extremely complex problem. The initial ambitious goal of implementing dynamic programming on a personal computer was not successful even with the state increment dynamic programming technique. The high dimensionality for the problem as formulated proved to be one of the greatest obstacles for this solution process.

The application of dynamic programming to the tracking problem does not require a sophisticated tracking algorithm to compute optimal controls. It does, however, require some performance criterion by which to measure the effect of the controls. This performance criteria, or penalty function, was somewhat hypothetical in this solution technique. As stated for this research effort, the penalty function attempted to minimize missile deviations from a desired position. It is conceivable that a CMMCA crew may have some other objective while tracking a missile. They may, for example, only want to minimize the time the cruise missile spends outside the radar sweep cone. The penalty function used for this research was completely subjective and other possible formulations exist.

The application of this model to the CMMCA tracking problem is limited in scope. As formulated, only single turns of the cruise missile can be simulated. The optimal

controls generated are a complex series of bank and thrust changes that a CMMCA aircrew might not be able to follow precisely. Because of these results, the model should only be used to gain insight into the necessary controls that might be required of CMMCA during a missile turn.

The model as formulated also proved to be extremely sensitive to the size of the time increment which defines the stages in the dynamic programming model. This investigation found that too small of an increment led to controls an CMMCA would not judge prudent while too large an increment can produce greater maximum deviations from the desired position. This result is likely caused by the form of the penalty function used in this investigation.

Recommendations

On the surface, dynamic programming is an excellent method of computing the controls necessary to track a missile test flight. The methodology accounts for the impact of the future position of the missile on control inputs at any stage. Three areas require further study before the methodology can be applied on a broader scale.

First, another investigation into the applicability of state increment dynamic programming still seems promising. The technique reduces the high-speed memory requirements and provides a method which allows the implementation of dynamic programming on a personal computer. This investigation points to the need for an accurate method to process

the transitions of block boundaries. The second technique proposed by Larson of extrapolating the penalty function into the not-yet-computed block may prove more accurate.

Another area for further investigation would be into the effect a linear interpolation has on the performance of the model. This investigation assumed a linear interpolation as a first cut. Perhaps a quadratic curve fitting technique or some other technique may lead to more accurate results.

Finally, and probably most importantly, a refinement of the penalty function for the methodology is needed. The penalty function used in this effort commands controls that keep the cruise missile near a desired position. It is possible, however, that this is not the true objective of a CMMCA aircrew. Currently, the CMMCA aircrew are also not certain as to their true objective. To evaluate the performance of different penalty functions, similar criteria to those used to evaluate the increment size for time in this investigation might be appropriate. Future applications of this methodology should investigate better definitions for this objective.

APPENDIX A. Equations of Motion

Equations of motion will apply to both the CMMCA and the cruise missile. The axis of the reference frame for the derivation of these equations will have the x-axis aligned with east and the y-axis aligned with true north. Define heading, h, as the angle between the y-axis and the velocity vector measured in a clockwise direction. The rate of change for the x position, y position, heading, and velocity can now be derived.

X and Y Position Rate of Change

When an aircraft makes a level turn, the velocity vector will change over time because of centripetal acceleration. The magnitude of the rate of change can be written

$$\left|\frac{dv}{dt}\right| = a_c. \tag{A.1}$$

It can be shown that the centripetal acceleration in a level turn is defined as follows:

$$a_c = g \tan(b) \tag{A.2}$$

where g is the magnitude of the acceleration vector due to gravity and b is the aircraft's bank angle (3:71). Combining equations A.1 and A.2 yields the relationship

$$\left|\frac{dv}{dt}\right| = g \tan(b). \tag{A.3}$$

When equation A.3 is integrated, with a speed v_o at t=0 and a constant bank angle, the following relationship results

$$|v(t)| = v_o + g \tan(b)\, t \qquad (A.4)$$

defining the magnitude of aircraft's velocity as a function of time.

By definition, the x-y components of the velocity vector define the rate of change of the aircraft's x and y position. To decompose equation A.4 into its x-y components, it can be thought of as the sum of two vector quantities.

The first part, v_o, is the aircraft's speed at time 0. When the aircraft is flying a heading h in degrees, the x-y components can be written

$$v_x = v_o \sin(h); \qquad (A.5)$$

$$v_y = v_o \cos(h). \qquad (A.6)$$

The second part, g tan(b) t, is the contribution of the centripetal acceleration to the velocity vector. Because the centripetal acceleration acts 90 degrees to the aircraft's heading, the x-y components can be written

$$v_x = g \tan(b) \sin(h + 90)\, t; \qquad (A.7)$$

$$v_y = g \tan(b) \cos(h + 90)\, t. \qquad (A.8)$$

Equations A 7 and A.8 can be further reduced to yield

$$v_x = g \tan(b) \cos(h)\, t; \qquad (A.9)$$

$$v_y = -g \tan(b) \sin(h)\, t. \qquad (A.10)$$

The direction for each of the above components is determined by the aircraft's bank angle as well as the heading. By defining a left bank as negative and a right bank as positive, the proper direction for each component will be obtained.

Bringing these two component parts together, the function defining the rate of change of the aircraft's x and y position can be written. The following relationships are obtained by combining the component parts described above into equation A.4:

$$\frac{dx}{dt} = v_o \sin(h) + g \tan(b) \cos(h)\, t \qquad (A.11)$$

and

$$\frac{dy}{dt} = v_o \cos(h) - g \tan(b) \sin(h)\, t. \qquad (A.12)$$

Heading Rate of Change

In a level turn, the rate of change of the airframe's heading is governed by the following (4:425):

$$\frac{dh}{dt} = \frac{g \tan(b)}{v_o} \qquad (A.13)$$

Velocity Rate of Change

For straight ahead acceleration, a summation of the forces acting along the longitudinal axis of the airframe results in the following equation:

$$\frac{dv}{dt} = \frac{(T - D)}{m} \qquad (A.14)$$

where T is the aircraft's thrust, D its drag, and m its mass. Each of these three parameters is a function of other aircraft parameters. Thrust is a function of altitude and fuel flow. Drag is a function of weight, velocity, altitude, and flight path angle. Mass is a function of the fuel flow.

Thus, the velocity rate of change is a complex relationship of many variables. For simplicity and the short time period under consideration, this research effort assumed nominal values of thrust, drag, and mass. This allowed accelerations to be considered constant.

APPENDIX B. Source Code

The source code for the FORTRAN program used to implement the dynamic program used to determine the optimal controls to track a cruise missile turn follows. The code was compiled and executed on a VAX running on the VMS operating system.

```
C ***************************************************************************
C *                                                                         *
C *  TITLE:      CMMCA Dynamic Program to Track Cruise Missile Manuevers    *
C *                                                                         *
C *  AUTHOR:     Capt Leonard G. Heavner                                    *
C *  DATE:       12 Jan 89                                                  *
C *                                                                         *
C *  DESCRIPTION:  This program implements a dynamic program algorithm     *
C *                to track a cruise missile during a turning manuever.     *
C *                The algorithm is implemented in three main modules.      *
C *                The first plots the cruise missile's position as it      *
C *                makes a turn.  The second module determines the          *
C *                optimal controls to track the missile at each            *
C *                quantized point in the state space.  The last module     *
C *                determines the optimal trajectory for a given            *
C *                inital state.  This is the controlling module.  It       *
C *                prompts the user through the use of a menu.  After       *
C *                the user makes a selection, the appropriate module       *
C *                is called.                                               *
C *                                                                         *
C ***************************************************************************
      PROGRAM CMMCA

      INCLUDE 'SYSVAR.COM/LIST'

          INTEGER     CHOICE

          LOGICAL     DONE,
     +                VALID
C
C  Initialize the program
C
          CALL INITPRM

          DONE = .FALSE.
          VALID = .TRUE.
C
C  Until the user is done, clear the screen and get request
C
   10     IF (.NOT.DONE) THEN
              CALL CLS
C
C  If choice is not valid, print error message
C
              IF (.NOT.VALID) THEN
                  PRINT 100, CHOICE
  100             FORMAT(1X,T26,'"',I1,'" IS NOT A VALID CHOICE'///)
                  VALID = .TRUE.
              ENDIF
C
C  Display menu and read choice
C
```

```
                PRINT 110
  110           FORMAT(1X,T25,'CMMCA FLIGHT PLAN GENERATOR'///
     +                    T20,'1 - ENTER CRUISE MISSILE FLIGHT PATH'//
     +                    T20,'2 - COMPUTE CMMCA FLIGHT PATH'//
     +                    T20,'3 - PRINT CMMCA FLIGHT PATH'//
     +                    T20,'4 - QUIT'///
     +                    T28,'    ENTER CHOICE'/)

                READ (*,*) CHOICE
C
C  Call the appropriate subroutine module or signal when done
C
                IF (CHOICE.EQ.1) THEN
                    CALL CMFTPN
                ELSEIF (CHOICE.EQ.2) THEN
                    CALL FLTPLN
                ELSEIF (CHOICE.EQ.3) THEN
                    CALL PRNTPP
                ELSEIF (CHOICE.EQ.4) THEN
                    DONE = .TRUE.
                ELSE
                    VALID = .FALSE.
                  ENDIF

                GOTO 10
              ENDIF
C
C  Clear the screen before quitting
C
            CALL CLS
          END
```

```
C ***********************************************************************
C *                                                                     *
C *  SUBROUTINE:        CMFTPN                                          *
C *                                                                     *
C *  DESCRIPTION:       Simulates the flight of a cruise missile in one *
C *                     turning manuever and records its position at    *
C *                     predetermined intervals.                        *
C *                                                                     *
C ***********************************************************************
      SUBROUTINE CMFTPN

      INCLUDE 'SYSVAR.COM'
      INCLUDE 'CMVAR.COM'

          INTEGER     NUMWP,  WAYPT,  I
C
C  Open the file to store the missile's position, clear the screen,
C       and initalize the variables
C
          OPEN (UNIT=10,FILE='CMPOSTN.DAT',ACCESS='DIRECT',
     +          STATUS='NEW',FORM='UNFORMATTED',RECL=8)

          CALL CLS
          TNOW = 0.
          ETA = 0.
          ETE = 0.
          TIMINC = .1
          G = 19.32
C
C  Display the header read the cruise missile's speed, starting
C       coordinates, and begin turn coordinates.
C
          PRINT 100
  100     FORMAT (1X,T25,'SUBROUTINE CRUISE MISSILE FLIGHT PATH'////
     +            1X,'ENTER CRUISE MISSILE SPEED (KNOTS) -'/)
          READ (*,*) SPEED
          SPEED = SPEED/60.0
          PRINT 110
  110     FORMAT (/1X,'ENTER THE STARTING COORDINATES (X AND Y) -'/)
          READ (*,*) XPOS,YPOS
          PRINT 120
  120     FORMAT (/1X,'ENTER THE BEGIN TURN COORDINATES (X AND Y) -'/)
          READ (*,*) NEXTX,NEXTY
C
C  Record this initial position and determine next time to record
C
          WRITE (10,REC=1) SNGL(XPOS), SNGL(YPOS)
          TRECRD = TIMINC
```

```
C
C  Move the missile to the begin turn coordinates in a linear fashion
C
           CALL LINEAR
C
C  Get the bank angle in the turn and the stopping coordinates
C
           PRINT 130
  130      FORMAT (/1X,'ENTER THE BANK ANGLE (DEGREES) -'/)
           READ (*,*) BANK
           BANK = BANK/180.0*PI

           PRINT 140
  140      FORMAT (/1X,'ENTER THE STOPPING COORDINATES (X AND Y) -'/)

           READ (*,*) NEXTX,NEXTY
C
C  Turn the missile. When headed toward the stopping coordinates,
C       move it again in a linear fashion until done
C
           CALL TURN
           CALL LINEAR
C
C  Determine the the last stage for the dynamic program.  The last
C       stage will be a whole increment of time.  Close the file
C       containing the cruise missile's position.
C
           TMAX = DTIME*IFIX(TNOW/DTIME)
           CLOSE (10)

           RETURN
          END
```

```
C ***************************************************************************
C *                                                                         *
C *  SUBROUTINE:        LINEAR                                              *
C *                                                                         *
C *  DESCRIPTION:       Moves the cruise missile in a linear fashion.       *
C *                     At the appropriate time, the missile's position     *
C *                     is recorded at the appropriate intervals.           *
C *                                                                         *
C ***************************************************************************
      SUBROUTINE LINEAR

      INCLUDE 'SYSVAR.COM'
      INCLUDE 'CMVAR.COM'

          DOUBLE PRECISION    RANGE
C
C  Determine the appropriate range and heading to the next point.
C      Also, calculate the x and y velocity components and the time
C      of arrival at this next point
C
          RANGE = DSQRT((NEXTX-XPOS)**2+(NEXTY-YPOS)**2)
          HEADNG = DATAN2(NEXTX-XPOS,NEXTY-YPOS)
          XSPEED = SPEED*DSIN(HEADNG)
          YSPEED = SPEED*DCOS(HEADNG)
          ETA = TNOW + SNGL(RANGE/SPEED)
C
C  While the missile has not arrived at the next point, move the
C      the missile forward to the next time to record position,
C      record position, and compute the next time to record
C
   10     IF (TRECRD.LE.ETA) THEN
              XPOS = XPOS+XSPEED*(TRECRD-TNOW)
              YPOS = YPOS+YSPEED*(TRECRD-TNOW)
              TNOW = TRECRD
              WRITE (10,REC=IFIX(TRECRD*RECRAT/DTIMF+1))
     +                                    SNGL(XPOS),SNGL(YPOS)
              TRECRD = TRECRD + TIMINC
              GOTO 10
            ENDIF
C
C  Upon arrival, ensure the missile is at the desired point
C
          TNOW = ETA
          XPOS = NEXTX
          YPOS = NEXTY

          RETURN
        END
```

```
C ***************************************************************************
C *                                                                         *
C *  SUBROUTINE:         TURN                                               *
C *                                                                         *
C *  DESCRIPTION:        Simulates the flight of the cruise missile in      *
C *                      a turn.  Continues the turn until the missile's    *
C *                      heading will take it to the next point.            *
C *                                                                         *
C ***************************************************************************
        SUBROUTINE TURN

        INCLUDE 'SYSVAR.COM'
        INCLUDE 'CMVAR.COM'

            LOGICAL     INTURN

            DOUBLE PRECISION    TRNRAT, STEPTM,
     +                          XSTEP,  YSTEP,
     +                          LASTBG, BEARNG
C
C  Initialize variables, determine turn rate, and determine the
C       bearing to the next point
C
            INTURN = .TRUE.
            TRNRAT = G*TAN(BANK)/SPEED
            LASTBG = DATAN2(NEXTX-XPOS,NEXTY-YPOS)
C
C  Make the correction to the missile's heading based on direction of
C       turn necessary to determine when the missile is headed at the
C       next point.
C
            IF (BANK.GT.0.AND.HEADNG.GT.LASTBG) THEN
                HEADNG = HEADNG - 2*DPI
              ELSEIF (BANK.LT.0.AND.HEADNG.LT.LASTBG) THEN
                HEADNG = HEADNG + 2*DPI
              ENDIF
C
C  While the missile is turning, do
C

   10       IF (INTURN) THEN
C
C  Determine the appropriate time increment to move the missile in
C       the turn
C
                IF (TRECRD-TNOW.LT.TIMINC/10) THEN
                    STEPTM = TRECRD-TNOW
                  ELSE
                    STEPTM = TIMINC/10
                  ENDIF
```

```
C
C  Determine the next heading, x and y position, and bearing to the
C       next point.
C
                HEADNG = HEADNG+TRNRAT*STEPTM
                XSTEP = XPOS+SPEED*DSIN(HEADNG)*STEPTM
                YSTEP = YPOS+SPEED*DCOS(HEADNG)*STEPTM
                BEARNG = DATAN2(NEXTX-XSTEP,NEXTY-YSTEP)
C
C  This logic determines when the missile transitions through the
C       third and fourth quadrants of the arctangent function.
C       If a transition is made, the missile's heading is adjusted
C       so that the missile will stop the turn at an appropriate time.
C
                IF (DSIGN(DPI,LASTBG).NE.DSIGN(DPI,BEARNG).AND.
     +                          DABS(BEARNG).GT.DPI/2) THEN
                   IF (LASTBG.GT.BEARNG.AND.HEADNG.GT.BEARNG) THEN
                        HEADNG = HEADNG - 2*DPI
                     ELSEIF (LASTBG.LT.BEARNG.AND.HEADNG.LT.BEARNG) THEN
                        HEADNG = HEADNG + 2*DPI
                     ENDIF
                  ENDIF
C
C  If the missile has turned past the next point, stop turn
C
                IF (BANK.GT.0.AND.HEADNG.GT.BEARNG) THEN
                     INTURN = .FALSE.
                  ELSEIF (BANK.LT.0.AND.HEADNG.LT.BEARNG) THEN
                     INTURN = .FALSE.
                  ENDIF
C
C  If the turn is not completed, update the missile's position, update
C       the time, and remember the current bearing to the next point
C
                IF (INTURN) THEN
                     XPOS = XSTEP
                     YPOS = YSTEP
                     TNOW = TNOW + STEPTM
                     LASTBG = BEARNG
```

```
C
C  If it is time for a recording, record the position and update the
C       next time to record.
C
                    IF (TNOW.EQ.TRECRD) THEN
                        WRITE (10,REC=IFIX(TRECRD*RECRAT/DTIME+1))
     +                                    SNGL(XPOS), SNGL(YPOS)
                        TRECRD = TRECRD + TIMINC
                      ENDIF
                  ENDIF
                GOTO 10
              ENDIF

            RETURN
           END
```

```
C ***************************************************************************
C *                                                                         *
C *  SUBROUTINE:        FLTPLN                                              *
C *                                                                         *
C *  DESCRIPTION:       This subroutine steps through each quantized        *
C *                     state and determines the optimal control for        *
C *                     each.  The optimal controls for all stages and      *
C *                     the final penalty at each quantized state are       *
C *                     saved in files.                                     *
C *                                                                         *
C ***************************************************************************
      SUBROUTINE FLTPLN

      INCLUDE 'SYSVAR.COM'
      INCLUDE 'DPVAR.COM'

          INTEGER     T,      I,      SAVCON(9000)
          REAL        LPNOW(9000),    LPLAS(9000)
C
C  Three 1-Dimensional arrays are equivalenced to the arrays holding
C       the penalties and optimal controls.  These arrays facilitate
C       the manipulation of the penalty and control arrays.
C
          EQUIVALENCE (SAVCON(1),OPTCNT(1,1,1,1)),
     +                (LPNOW(1),PENNOW(1,1,1,1)),
     +                (LPLAS(1),PENLAS(1,1,1,1))
C
C  Open the file containing the cruise missile position.  Also,
C       establish the files to contain optimal control and final
C       penalties.
C
          OPEN (UNIT=10,FILE='CMPOSTN.DAT',ACCESS='DIRECT',
     +          STATUS='OLD',FORM='UNFORMATTED',RECL=8)
          OPEN (UNIT=11,FILE='PENALTY.DAT',ACCESS='SEQUENTIAL',
     +          STATUS='NEW')
          OPEN (UNIT=12,FILE='CONTROL.DAT',ACCESS='DIRECT',
     +          STATUS='NEW',FORM='UNFORMATTED',RECL=6000)
C
C  Clear the screen and display the header
C
          CALL CLS
          PRINT *,'SUBROUTINE COMPUTE CMMCA FLIGHT PATH'
          PRINT *
          PRINT *
C
C  Initialize the dynamic program variables and display the last time
C       being called a stage
C
          CALL INTDP
          PRINT *,'COMPUTING PENALTY AT TMAX =',TMAX
          PRINT *
```

```
C
C  Read the missile position at the last time stage and display on
C       screen
C
            READ (10,REC=IFIX(TMAX*RECRAT/DTIME+1)) CMXPOS, CMYPOS
            PRINT *,'CMXPOS =',CMXPOS,'  CMYPOS =',CMYPOS
C
C  For each quantized state, determine the penalty for arriving
C       at that state in the last stage
C
            DO 10 X=1,WX
              PRINT *,'X BLOCK =',X
              XPOS = XMIN+(X-1)*DX
              DO 20 Y=1,WY
                YPOS = YMIN+(Y-1)*DY
                DO 30 HDG=1,WHDG
                  HDGPOS = HDGMIN+(HDG-1)*DHDG
                  DO 40 VEL=1,WVEL
                    VELPOS = VELMIN+(VEL-1)*DVEL
                    PENLAS(X,Y,HDG,VEL) =
     +              PENLTY(XPOS,YPOS,HDGPOS,0.0,0.0,CMXPOS,CMYPOS,DTIME)
   40             CONTINUE
   30           CONTINUE
   20         CONTINUE
   10       CONTINUE
C
C  For each of the remaining stages, display the time for the stage,
C       read and display the missile position at that time
C
            DO 50 T=IFIX(TMAX/DTIME),1,-1
            PRINT *
            PRINT *,'COMPUTING AT TIME =',T*DTIME-DTIME
            PRINT *
              READ (10,REC=(T-1)*RECRAT+1) CMXPOS, CMYPOS
              PRINT *,'CMXPOS =',CMXPOS,'  CMYPOS =',CMYPOS
```

```
C
C  For each quantized state, call the subroutine that determines the
C       optimal control for the state and missile position
C
              DO 60 X=1,WX
                PRINT *,'X BLOCK =',X
                XPOS = XMIN+(X-1)*DX
                DO 70 Y=1,WY
                  YPOS = YMIN+(Y-1)*DY
                  DO 80 HDG=1,WHDG
                    HDGPOS = HDGMIN+(HDG-1)*DHDG
                    DO 90 VEL=1,WVEL
                      VELPOS = VELMIN+(VEL-1)*DVEL
                      CALL CHKCNT
                      OPTCNT(X,Y,HDG,VEL) = COPT
   90                 CONTINUE
   80               CONTINUE
   70             CONTINUE
   60           CONTINUE
C
C  Save the optimal controls for this stage after all quantized states
C       are processed
C
              WRITE (12,REC=(T*6)-5) (SAVCON(I),I=1,1500)
              WRITE (12,REC=(T*6)-4) (SAVCON(I),I=1501,3000)
              WRITE (12,REC=(T*6)-3) (SAVCON(I),I=3001,4500)
              WRITE (12,REC=(T*6)-2) (SAVCON(I),I=4501,6000)
              WRITE (12,REC=(T*6)-1) (SAVCON(I),I=6001,7500)
              WRITE (12,REC=(T*6))   (SAVCON(I),I=7501,9000)
C
C  The current penalties become the penalties of the last stage in
C       preparation of processing the next time stage
C
              DO 100 I=1,9000
                LPLAS(I) = LPNOW(I)
  100           CONTINUE
   50         CONTINUE
C
C  Save the last penalties in the proper file
C
              WRITE (11,*) (LPNOW(I),I=1,9000)
C
C  Close all the files
C
  550         CLOSE (10)
              CLOSE (11)
              CLOSE (12)

              RETURN
            END
```

```
C ***************************************************************************
C *                                                                         *
C *  SUBROUTINE:        CHKCNT                                              *
C *                                                                         *
C *  DESCRIPTION:       This subroutine determines the optimal control      *
C *                     for the current stage and quantized state.          *
C *                                                                         *
C ***************************************************************************
      SUBROUTINE CHKCNT

      INCLUDE 'SYSVAR.COM'
      INCLUDE 'DPVAR.COM'

          REAL        NXTX,   NXTY,   NXTHDG, NXTVEL,
     +                LOCPEN, NXTPEN,
     +                PENS(16)
          INTEGER     I,      J,      K,      L,      N,
     +                XBAS,   YBAS,   HDGBAS, VELBAS, HADJ
C
C  Initialize the optimal control to zero and compute the penalty for
C       arriving at the current state
C
          COPT = 0
          LOCPEN = PENLTY(XPOS,YPOS,HDGPOS,CONTRL(C,1),
     +                                CONTRL(C,2),CMXPOS,CMYPOS,DTIME)
C
C  For each control, determine the state of the next stage for applying
C       that control
C
          DO 10 C=1,5
              NXTX = XPOS + XCHG(HDG,VEL,C)
              NXTY = YPOS + YCHG(HDG,VEL,C)
              NXTHDG = HDGPOS + HDGCHG(HDG,VEL,C)
              NXTVEL = VELPOS + VELCHG(C)
C
C  If the next state is within limits, continue
C
              IF (NXTX.GE.XMIN.AND.NXTX.LE.XMAX.AND.
     +                NXTY.GE.YMIN.AND.NXTY.LE.YMAX.AND.
     +                NXTVEL.GE.VELMIN.AND.NXTVEL.LE.VELMAX) THEN
C
C  Determine the nearest quantized state to the state that will be
C       reached if the control is applied
C
                  XBAS = IFIX((NXTX-XMIN)/DX+1)
                  YBAS = IFIX((NXTY-YMIN)/DY+1)
                  HDGBAS = IFIX((NXTHDG-HDGMIN)/DHDG+1)
                  VELBAS = IFIX((NXTVEL-VELMIN)/DVEL+1)
```

```
C
C  This code ensures that the interpolation will not try to interpolate
C       from a point not within limits of the state constraints
C
                    IF (XBAS.EQ.WX) XBAS = WX - 1
                    IF (YBAS.EQ.WY) YBAS = WY - 1
                    IF (HDGBAS.EQ.WHDG) THEN
                        HADJ = 1-WHDG
                      ELSE
                        HADJ = 1
                      ENDIF
                    IF (VELBAS.EQ.WVEL) VELBAS = WVEL - 1
C
C  Load the array used for interpolation.
C
                    PENS(1)=PENLAS(XBAS,YBAS,HDGBAS,VELBAS)
                    PENS(2)=PENLAS(XBAS,YBAS,HDGBAS,VELBAS+1)
                    PENS(3)=PENLAS(XBAS,YBAS,HDGBAS+HADJ,VELBAS)
                    PENS(4)=PENLAS(XBAS,YBAS,HDGBAS+HADJ,VELBAS+1)
                    PENS(5)=PENLAS(XBAS,YBAS+1,HDGBAS,VELBAS)
                    PENS(6)=PENLAS(XBAS,YBAS+1,HDGBAS,VELBAS+1)
                    PENS(7)=PENLAS(XBAS,YBAS+1,HDGBAS+HADJ,VELBAS)
                    PENS(8)=PENLAS(XBAS,YBAS+1,HDGBAS+HADJ,VELBAS+1)
                    PENS(9)=PENLAS(XBAS+1,YBAS,HDGBAS,VELBAS)
                    PENS(10)=PENLAS(XBAS+1,YBAS,HDGBAS,VELBAS+1)
                    PENS(11)=PENLAS(XBAS+1,YBAS,HDGBAS+HADJ,VELBAS)
                    PENS(12)=PENLAS(XBAS+1,YBAS,HDGBAS+HADJ,VELBAS+1)
                    PENS(13)=PENLAS(XBAS+1,YBAS+1,HDGBAS,VELBAS)
                    PENS(14)=PENLAS(XBAS+1,YBAS+1,HDGBAS,VELBAS+1)
                    PENS(15)=PENLAS(XBAS+1,YBAS+1,HDGBAS+HADJ,VELBAS)
                    PENS(16)=PENLAS(XBAS+1,YBAS+1,HDGBAS+HADJ,VELBAS+1)
C
C  Determine the difference between the next state and the nearest
C       quantized state
C
                    DIFF(1) = NXTVEL - (VELMIN+(VELBAS-1)*DVEL)
                    DIFF(2) = NXTHDG - (HDGMIN+(HDGBAS-1)*DHDG)
                    DIFF(3) = NXTY - (YMIN+(YBAS-1)*DY)
                    DIFF(4) = NXTX - (XMIN+(XBAS-1)*DX)
C
C  Interpolate the penalty for arriving at the next state
C
                    NXTPEN = EST(DIFF,DPOS,PENS)
C
C  If no allowable control found so far, make this the optimal control
C       to this point and save the penalty
C
                    IF (COPT.EQ.0) THEN
                        PENNOW(X,Y,HDG,VEL) = LOCPEN + NXTPEN
                        COPT = C
```

```
C
C  Else if, the total penalty for this control is less than any previous
C       control, make it the optimal control and save the penalty
C                       ELSEIF (LOCPEN+NXTPEN.LT.PENNOW(X,Y,HDG,VEL)) THEN
                          PENNOW(X,Y,HDG,VEL) = LOCPEN + NXTPEN
                          COPT = C
                        ENDIF
                      ENDIF
   10           CONTINUE
C
C  If no optimal controls found, make the penalty some big number
C
              IF (COPT.EQ.0) PENNOW(X,Y,HDG,VEL)=BIGNUM

              RETURN
            END
```

```
C ***************************************************************************
C *                                                                         *
C *  SUBROUTINE:         PRNTPP                                             *
C *                                                                         *
C *  DESCRIPTION:        This subroutine retreives the optimal controls     *
C *                      and simulates the flight of CMMCA as it tracks     *
C *                      the cruise missile.  The user is given two         *
C *                      options of starting this revrieval.  The           *
C *                      initial of CMMCA can be explicitely stated or      *
C *                      the optimal starting position can be found.        *
C *                      results of the simulation are displayed to the     *
C *                      screen and saved in the file RESULTS.DAT.          *
C *                                                                         *
C ***************************************************************************
      SUBROUTINE PRNTPP

          INCLUDE 'SYSVAR.COM'
          INCLUDE 'DPVAR.COM'

          INTEGER      OPTX,   OPTY,   OPTHDG, OPTVEL, I, J,
     +                 XBAS,   YBAS,   HDGBAS, VELBAS, HADJ,
     +                 SAVCON(9000),   CHOICE
          REAL         LPNOW(9000),    LPLAS(9000),
     +                 BANK,   THRUST, TEMP(16),       ROPT,
     +                 RANGE,  AZ,     PHDG,   ALT,    DEV
          DOUBLE PRECISION    NXTX,   NXTY,   NXTHDG, NXTVEL
C
C  Two 1-Dimensional arrays are equivalenced to the optimal control and
C       penalty arrays for ease of manipulation.
C
          EQUIVALENCE (SAVCON(1),OPTCNT(1,1,1,1)),
     +                (LPNOW(1),PENNOW(1,1,1,1))
C
C  Open the files containing the missile position, the optimal controls
C       at each stage, and the final penalties.  Also, initialize the
C       file which will contain the results of the simulation of the
C       flight.
C
          OPEN (UNIT=10,FILE='CMPOSTN.DAT',ACCESS='DIRECT',
     +          STATUS='OLD',FORM='UNFORMATTED',RECL=8)
          OPEN (UNIT=11,FILE='PENALTY.DAT',ACCESS='SEQUENTIAL',
     +          STATUS='OLD')
          OPEN (UNIT=12,FILE='CONTROL.DAT',ACCESS='DIRECT',
     +          STATUS='OLD',FORM='UNFORMATTED',RECL=6000)
          OPEN (UNIT=13,FILE='RESULTS.DAT',ACCESS='SEQUENTIAL',
     +          STATUS='NEW')
```

```
C
C  Clear the screen, display the menu and read the user's choice
C
           CALL CLS
           PRINT 200
  200      FORMAT (1X,T15,'SUBROUTINE DISPLAY CMMCA FLIGHT PATH'////
     +             1X,T16,'1 - SET CMMCA STARTING CONDITIONS'//
     +             1X,T16,'2 - DP OPTIMAL SOLUTION'///
     +             1X,T20,'ENTER CHOICE'/)
           READ (*,*) CHOICE
C
C  Again, clear the screen and display a working message
C
           CALL CLS
           PRINT *,'WORKING'
           PRINT *
C
C  Read the penalties of the last stage and initialize the altitude of
C       of the simulation.  Also, initialize the array containing the
C       size of the increments between the quantized states.
C
           READ (11,*) (LPNOW(I),I=1,9000)
           ROPT = 15.0
           ALT = 29000/6076
           DPOS(1) = DVEL
           DPOS(2) = DHDG
           DPOS(3) = DY
           DPOS(4) = DX
C
C  Read the set of optimals controls for the first stage
C
           READ (12,REC=1) (SAVCON(I),I=1,1500)
           READ (12,REC=2) (SAVCON(I),I=1501,3000)
           READ (12,REC=3) (SAVCON(I),I=3001,4500)
           READ (12,REC=4) (SAVCON(I),I=4501,6000)
           READ (12,REC=5) (SAVCON(I),I=6001,7500)
           READ (12,REC=6) (SAVCON(I),I=7501,9000)
```

```
C
C  If the choice is to state the CMMCA initial state, clear the screen
C       and request starting position, heading, and airspeed.  Convert
C       the airspeed to nm/min and heading to radians.
C
            IF (CHOICE.EQ.1) THEN
                CALL CLS

                PRINT 205
  205           FORMAT (///1X,'ENTER CMMCA STARTING POINT (X AND Y) -'/)
                READ (*,*)NXTX,NXTY

                PRINT 210
  210           FORMAT (///1X,'ENTER INITIAL HEADING (DEGREES) -'/)
                READ (*,*) NXTHDG
                IF (NXTHDG.GT.180.0) NXTHDG = NXTHDG - 360.0
                NXTHDG = NXTHDG/180.0*DPI

                PRINT 215
  215           FORMAT (///1X,'ENTER INITIAL VELOCITY (KNOTS) -'/)
                READ (*,*) NXTVEL
                NXTVEL = NXTVEL/60.0
C
C  Find the nearest quantized state
C
                XBAS = IFIX((SNGL(NXTX)-XMIN)/DX+1)
                YBAS = IFIX((SNGL(NXTY)-YMIN)/DY+1)
                HDGBAS = IFIX((SNGL(NXTHDG)-HDGMIN)/DHDG+1)
                VELBAS = IFIX((SNGL(NXTVEL)-VELMIN)/DVEL+1)

                IF (XBAS.EQ.WX) XBAS = WX-1
                IF (YBAS.EQ.WY) YBAS = WY-1
                IF (HDGBAS.EQ.WHDG) THEN
                    HADJ = 1-WHDG
                  ELSE
                    HADJ = 1
                  ENDIF
                IF (VELBAS.EQ.WVEL) VELBAS = WVEL-1
```

```
C
C  Load the array for interpolation of the optimal bank angle
C
          TEMP(1)=CONTRL(OPTCNT(XBAS,YBAS,HDGBAS,VELBAS),1)
          TEMP(2)=CONTRL(OPTCNT(XBAS,YBAS,HDGBAS,VELBAS+1),1)
          TEMP(3)=CONTRL(OPTCNT(XBAS,YBAS,HDGBAS+HADJ,VELBAS),1)
          TEMP(4)=CONTRL(OPTCNT(XBAS,YBAS,HDGBAS+HADJ,VELBAS+1),1)
          TEMP(5)=CONTRL(OPTCNT(XBAS,YBAS+1,HDGBAS,VELBAS),1)
          TEMP(6)=CONTRL(OPTCNT(XBAS,YBAS+1,HDGBAS,VELBAS+1),1)
          TEMP(7)=CONTRL(OPTCNT(XBAS,YBAS+1,HDGBAS+HADJ,VELBAS),1)
          TEMP(8)=CONTRL(OPTCNT(XBAS,YBAS+1,HDGBAS+HADJ,VELBAS+1),1)
          TEMP(9)=CONTRL(OPTCNT(XBAS+1,YBAS,HDGBAS,VELBAS),1)
          TEMP(10)=CONTRL(OPTCNT(XBAS+1,YBAS,HDGBAS,VELBAS+1),1)
          TEMP(11)=CONTRL(OPTCNT(XBAS+1,YBAS,HDGBAS+HADJ,VELBAS),1)
          TEMP(12)=CONTRL(OPTCNT(XBAS+1,YBAS,HDGBAS+HADJ,VELBAS+1),1)
          TEMP(13)=CONTRL(OPTCNT(XBAS+1,YBAS+1,HDGBAS,VELBAS),1)
          TEMP(14)=CONTRL(OPTCNT(XBAS+1,YBAS+1,HDGBAS,VELBAS+1),1)
          TEMP(15)=CONTRL(OPTCNT(XBAS+1,YBAS+1,HDGBAS+HADJ,VELBAS),1)
          TEMP(16)=CONTRL(OPTCNT(XBAS+1,YLAS+1,HDGBAS+HADJ,VELBAS+1),1)
C
C  Compute the difference between the requested state and the nearest
C       quantized state.  Then interpolate the optimal bank angle.
C
               DIFF(1) = SNGL(NXTVEL) - (VELMIN+(VELBAS-1)*DVEL)
               DIFF(2) = SNGL(NXTHDG) - (HDGMIN+(HDGBAS-1)*DHDG)
               DIFF(3) = SNGL(NXTY) - (YMIN+(YBAS-1)*DY)
               DIFF(4) = SNGL(NXTX) - (XMIN+(XBAS-1)*DX)

               BANK = EST(DIFF,DPOS,TEMP)
C
C  Load the array for the interpolation of the optimal thrust control.
C       Then interpolate the optimal thrust control.
C
          TEMP(1)=CONTRL(OPTCNT(XBAS,YBAS,HDGBAS,VELBAS),2)
          TEMP(2)=CONTRL(OPTCNT(XBAS,YBAS,HDGBAS,VELBAS+1),2)
          TEMP(3)=CONTRL(OPTCNT(XBAS,YBAS,HDGBAS+HADJ,VELBAS),2)
          TEMP(4)=CONTRL(OPTCNT(XBAS,YBAS,HDGBAS+HADJ,VELBAS+1),2)
          TEMP(5)=CONTRL(OPTCNT(XBAS,YBAS+1,HDGBAS,VELBAS),2)
          TEMP(6)=CONTRL(OPTCNT(XBAS,YBAS+1,HDGBAS,VELBAS+1),2)
          TEMP(7)=CONTRL(OPTCNT(XBAS,YBAS+1,HDGBAS+HADJ,VELBAS),2)
          TEMP(8)=CONTRL(OPTCNT(XBAS,YBAS+1,HDGBAS+HADJ,VELBAS+1),2)
          TEMP(9)=CONTRL(OPTCNT(XBAS+1,YBAS,HDGBAS,VELBAS),2)
          TEMP(10)=CONTRL(OPTCNT(XBAS+1,YBAS,HDGBAS,VELBAS+1),2)
          TEMP(11)=CONTRL(OPTCNT(XBAS+1,YBAS,HDGBAS+HADJ,VELBAS),2)
          TEMP(12)=CONTRL(OPTCNT(XBAS+1,YBAS,HDGBAS+HADJ,VELBAS+1),2)
          TEMP(13)=CONTRL(OPTCNT(XBAS+1,YBAS+1,HDGBAS,VELBAS),2)
          TEMP(14)=CONTRL(OPTCNT(XBAS+1,YBAS+1,HDGBAS,VELBAS+1),2)
          TEMP(15)=CONTRL(OPTCNT(XBAS+1,YBAS+1,HDGBAS+HADJ,VELBAS),2)
          TEMP(16)=CONTRL(OPTCNT(XBAS+1,YBAS+1,HDGBAS+HADJ,VELBAS+1),2)

               THRUST = EST(DIFF,DPOS,TEMP)
```

```
C
C  Else determine the quantized state that results in the least penalty
C
               ELSE
                 OPTX=1
                 OPTY=1
                 OPTHDG=1
                 OPTVEL=1
                 DO 20 X=1,WX
                     DO 30 Y=1,WY
                         DO 40 HDG=1,WHDG
                             DO 50 VEL=1,WVEL
                                 IF (PENNOW(X,Y,HDG,VEL).LT.
     +                              PENNOW(OPTX,OPTY,OPTHDG,OPTVEL)) THEN

                                     OPTX = X
                                     OPTY = Y
                                     OPTHDG = HDG
                                     OPTVEL = VEL
                                   ENDIF
    50                           CONTINUE
    40                       CONTINUE
    30                   CONTINUE
    20               CONTINUE
C
C  Establish the starting position and optimal controls
C
                 NXTX = DBLE(XMIN + (OPTX-1)*DX)
                 NXTY = DBLE(YMIN + (OPTY-1)*DY)
                 NXTHDG = DBLE(HDGMIN + (OPTHDG-1)*DHDG)
                 NXTVEL = DBLE(VELMIN + (OPTVEL-1)*DVEL)

                 BANK   = CONTRL(OPTCNT(OPTX,OPTY,OPTHDG,OPTVEL),1)
                 THRUST = CONTRL(OPTCNT(OPTX,OPTY,OPTHDG,OPTVEL),2)
               ENDIF
C
C  Read the first missile position, and calculate the parameters to
C       be displayed on the screen.  Calculate the initial deviation.
C
             READ (10,REC=1) CMXPOS,CMYPOS

             PHDG = NXTHDG*180.0/PI
             IF (PHDG.LT.0) PHDG = PHDG + 360.0
             RANGE = SQRT((NXTX-CMXPOS)**2+(NXTY-CMYPOS)**2+ALT**2)
             AZ = ATAN2(CMXPOS-NXTX,CMYPOS-NXTY) - NXTHDG
             IF (AZ.GT.PI) AZ = AZ - 2*PI
             IF (AZ.LT.-PI) AZ = AZ + 2*PI
             DEV = (ABS(ROPT-RANGE)+ABS(RANGE*AZ))*DTIME

             CALL CLS
```

```
C
C  Display the header and the initial conditions
C
           WRITE (*,100)
           WRITE (13,100)
  100      FORMAT (1X,'          CMMCA PARAMETERS          CONTROLS  ',
     +             '   MISSILE POS      PERFORMANCE'/
     +             1X,'TNOW   XPOS    YPOS   HDG   VEL   BANK  THRUST',
     +             '   CMX     CMY    RANGE   AZIMUTH'/)
           WRITE (*,105) TEMP(1)*0.0,NXTX,NXTY,PHDG,NXTVEL*60.0,
     +                   BANK*180.0/PI,THRUST, CMXPOS, CMYPOS,
     +                   RANGE,AZ*180.0/PI
           WRITE (13,105) TEMP(1)*0.0,NXTX,NXTY,PHDG,NXTVEL*60.0,
     +                   BANK*180.0/PI,THRUST, CMXPOS, CMYPOS,
     +                   RANGE,AZ*180.0/PI
  105      FORMAT (1X,F4.2,2X,F6.3,2X,F6.3,2X,F4.0,2X,F4.0,2X,F5.1,2X,
     +            F5.2,2X,F6.3,2X,F6.3,2X,F5.2,2X,F7.2)
C
C  For each stage, apply the optimal controls
C
           DO 60 J=2,IFIX(TMAX/DTIME+1)
              CALL NEWPOS(NXTX,NXTY,NXTHDG,NXTVEL,BANK,THRUST,
     +                                              DTIME/RECRAT)
C
C  For each of the intermediate missile positions in the stage,
C       simulate flight and compute for display the performance
C
              DO 65 K=1,RECRAT-1
                 READ (10,REC=(J-2)*RECRAT+K+1)CMXPOS,CMYPOS
                 PHDG = SNGL(NXTHDG)*180.0/PI
                 IF (PHDG.LT.0) PHDG = PHDG + 360.0
                 RANGE = SQRT((SNGL(NXTX)-CMXPOS)**2+
     +                              (SNGL(NXTY)-CMYPOS)**2+ALT**2)
                 AZ = ATAN2(CMXPOS-SNGL(NXTX),CMYPOS-SNGL(NXTY))-
     +                                              SNGL(NXTHDG)
                 IF (AZ.GT.PI) AZ = AZ - 2*PI
                 IF (AZ.LT.-PI) AZ = AZ + 2*PI
                 DEV = DEV + (ABS(ROPT-RANGE)+ABS(RANGE*AZ))*DTIME
                 WRITE (*,105) (J-2)*DTIME+K*DTIME/RECRAT,NXTX,NXTY,
     +                         PHDG,NXTVEL*60.0,BANK*180.0/PI,THRUST,
     +                         CMXPOS,CMYPOS,RANGE,AZ*180.0/PI
                 WRITE (13,105) (J-2)*DTIME+K*DTIME/RECRAT,NXTX,NXTY,
     +                         PHDG,NXTVEL*60.0,BANK*180.0/PI,THRUST,
     +                         CMXPOS,CMYPOS,RANGE,AZ*180.0/PI

                 CALL NEWPOS(NXTX,NXTY,NXTHDG,NXTVEL,BANK,THRUST,
     +                                              DTIME/RECRAT)
   65         CONTINUE
```

```
C
C  Read the next missile position and compute performance parameters
C
                READ (10,REC=(J-1)*RECRAT+1) CMXPOS,CMYPOS

                PHDG = SNGL(NXTHDG)*180.0/PI
                IF (PHDG.LT.0) PHDG = PHDG + 360.0
                RANGE = SQRT((SNGL(NXTX)-CMXPOS)**2+
     +                                  (SNGL(NXTY)-CMYPOS)**2+ALT**2)
                AZ = ATAN2(CMXPOS-SNGL(NXTX),CMYPOS-SNGL(NXTY))-
     +                                                  SNGL(NXTHDG)
                IF (AZ.GT.PI) AZ = AZ - 2*PI
                IF (AZ.LT.-PI) AZ = AZ + 2*PI
                DEV = DEV + (ABS(ROPT-RANGE)+ABS(RANGE*AZ))*DTIME
C
C  If this is not the last stage, get the next set of optimal controls
C       and interpolate the next optimal control in the same procedure
C       as above
C
                IF (J.LT.IFIX(TMAX/DTIME+1)) THEN
                    READ (12,REC=(J*6)-5) (SAVCON(I),I=1,1500)
                    READ (12,REC=(J*6)-4) (SAVCON(I),I=1501,3000)
                    READ (12,REC=(J*6)-3) (SAVCON(I),I=3001,4500)
                    READ (12,REC=(J*6)-2) (SAVCON(I),I=4501,6000)
                    READ (12,REC=(J*6)-1) (SAVCON(I),I=6001,7500)
                    READ (12,REC=(J*6)) (SAVCON(I),I=7501,9000)

                    XBAS = IFIX((SNGL(NXTX)-XMIN)/DX+1)
                    YBAS = IFIX((SNGL(NXTY)-YMIN)/DY+1)
                    HDGBAS = IFIX((SNGL(NXTHDG)-HDGMIN)/DHDG+1)
                    VELBAS = IFIX((SNGL(NXTVEL)-VELMIN)/DVEL+1)

                    IF (XBAS.EQ.WX) XBAS = WX-1
                    IF (YBAS.EQ.WY) YBAS = WY-1
                    IF (HDGBAS.EQ.WHDG) THEN
                        HADJ = 1-WHDG
                      ELSE
                        HADJ = 1
                      ENDIF
                    IF (VELBAS.EQ.WVEL) VELBAS = WVEL-1
```

```
            TEMP(1)=CONTRL(OPTCNT(XBAS,YBAS,HDGBAS,VELBAS),1)
            TEMP(2)=CONTRL(OPTCNT(XBAS,YBAS,HDGBAS,VELBAS+1),1)
            TEMP(3)=CONTRL(OPTCNT(XBAS,YBAS,HDGBAS+HADJ,VELBAS),1)
            TEMP(4)=CONTRL(OPTCNT(XBAS,YBAS,HDGBAS+HADJ,VELBAS+1),1)
            TEMP(5)=CONTRL(OPTCNT(XBAS,YBAS+1,HDGBAS,VELBAS),1)
            TEMP(6)=CONTRL(OPTCNT(XBAS,YBAS+1,HDGBAS,VELBAS+1),1)
            TEMP(7)=CONTRL(OPTCNT(XBAS,YBAS+1,HDGBAS+HADJ,VELBAS),1)
            TEMP(8)=CONTRL(OPTCNT(XBAS,YBAS+1,HDGBAS+HADJ,VELBAS+1),1)
            TEMP(9)=CONTRL(OPTCNT(XBAS+1,YBAS,HDGBAS,VELBAS),1)
            TEMP(10)=CONTRL(OPTCNT(XBAS+1,YBAS,HDGBAS,VELBAS+1),1)
            TEMP(11)=CONTRL(OPTCNT(XBAS+1,YBAS,HDGBAS+HADJ,VELBAS),1)
            TEMP(12)=CONTRL(OPTCNT(XBAS+1,YBAS,HDGBAS+HADJ,VELBAS+1),1)
            TEMP(13)=CONTRL(OPTCNT(XBAS+1,YBAS+1,HDGBAS,VELBAS),1)
            TEMP(14)=CONTRL(OPTCNT(XBAS+1,YBAS+1,HDGBAS,VELBAS+1),1)
            TEMP(15)=CONTRL(OPTCNT(XBAS+1,YBAS+1,HDGBAS+HADJ,VELBAS),1)
            TEMP(16)=CONTRL(OPTCNT(XBAS+1,YBAS+1,HDGBAS+HADJ,VELBAS+1),1)

                    DIFF(1) = SNGL(NXTVEL) - (VELMIN+(VELBAS-1)*DVEL)
                    DIFF(2) = SNGL(NXTHDG) - (HDGMIN+(HDGBAS-1)*DHDG)
                    DIFF(3) = SNGL(NXTY) - (YMIN+(YBAS-1)*DY)
                    DIFF(4) = SNGL(NXTX) - (XMIN+(XBAS-1)*DX)

                    BANK = EST(DIFF,DPOS,TEMP)

            TEMP(1)=CONTRL(OPTCNT(XBAS,YBAS,HDGBAS,VELBAS),2)
            TEMP(2)=CONTRL(OPTCNT(XBAS,YBAS,HDGBAS,VELBAS+1),2)
            TEMP(3)=CONTRL(OPTCNT(XBAS,YBAS,HDGBAS+HADJ,VELBAS),2)
            TEMP(4)=CONTRL(OPTCNT(XBAS,YBAS,HDGBAS+HADJ,VELBAS+1),2)
            TEMP(5)=CONTRL(OPTCNT(XBAS,YBAS+1,HDGBAS,VELBAS),2)
            TEMP(6)=CONTRL(OPTCNT(XBAS,YBAS+1,HDGBAS,VELBAS+1),2)
            TEMP(7)=CONTRL(OPTCNT(XBAS,YBAS+1,HDGBAS+HADJ,VELBAS),2)
            TEMP(8)=CONTRL(OPTCNT(XBAS,YBAS+1,HDGBAS+HADJ,VELBAS+1),2)
            TEMP(9)=CONTRL(OPTCNT(XBAS+1,YBAS,HDGBAS,VELBAS),2)
            TEMP(10)=CONTRL(OPTCNT(XBAS+1,YBAS,HDGBAS,VELBAS+1),2)
            TEMP(11)=CONTRL(OPTCNT(XBAS+1,YBAS,HDGBAS+HADJ,VELBAS),2)
            TEMP(12)=CONTRL(OPTCNT(XBAS+1,YBAS,HDGBAS+HADJ,VELBAS+1),2)
            TEMP(13)=CONTRL(OPTCNT(XBAS+1,YBAS+1,HDGBAS,VELBAS),2)
            TEMP(14)=CONTRL(OPTCNT(XBAS+1,YBAS+1,HDGBAS,VELBAS+1),2)
            TEMP(15)=CONTRL(OPTCNT(XBAS+1,YBAS+1,HDGBAS+HADJ,VELBAS),2)
            TEMP(16)=CONTRL(OPTCNT(XBAS+1,YBAS+1,HDGBAS+HADJ,VELBAS+1),2)

                    THRUST = EST(DIFF,DPOS,TEMP)
C
C  Display the results
C
                    WRITE (*,105) (J-1)*DTIME,NXTX,NXTY,PHDG,
     +                                  NXTVEL*60.0,BANK*180.0/PI,THRUST,
     +                                  CMXPOS,CMYPOS,RANGE,AZ*180.0/PI
                    WRITE (13,105) (J-1)*DTIME,NXTX,NXTY,PHDG,
     +                                  NXTVEL*60.0,BANK*180.0/PI,THRUST,
     +                                  CMXPOS,CMYPOS,RANGE,AZ*180.0/PI
```

```
C
C  Else the last stage has been reached.  Display on the final position
C       and performance parameters along with average deviations
C
                  ELSE
                    WRITE (*,110) (J-1)*DTIME,NXTX,NXTY,PHDG,
     +                             NXTVEL*60.0,CMXPOS,CMYPOS,
     +                             RANGE,AZ*180.0/PI
                    WRITE (13,110) (J-1)*DTIME,NXTX,NXTY,PHDG,
     +                             NXTVEL*60.0,CMXPOS,CMYPOS,
     +                             RANGE,AZ*180.0/PI
  110       FORMAT (1X,F4.2,2X,F6.3,2X,F6.3,2X,F4.0,2X,F4.0,16X,
     +              F6.3,2X,F6.3,2X,F5.2,2X,F7.2)

                    PRINT *
                    PRINT *,'AVERAGE TIME WEIGHTED DEVIATIONS FROM',
     +                      'DESIRED POSITION =',DEV/TMAX
                    WRITE (13,*)
                    WRITE (13,*) 'AVERAGE TIME WEIGHTED DEVIATIONS FROM',
     +                      'DESIRED POSITION =',DEV/TMAX
                  ENDIF

   60           CONTINUE
C
C  Hold program execution until the user is ready to continue
C
              PRINT *
              PRINT *
              PRINT *,'PRESS <ENTER> TO CONTINUE'
              READ (*,*)
C
C  Close all files
C
              CLOSE (10)
              CLOSE (11)
              CLOSE (12)
              CLOSE (13)

              RETURN
            END
```

```
C ***************************************************************************
C *                                                                         *
C *  SUBROUTINE:        INTPRM                                              *
C *                                                                         *
C *  DESCRIPTION:       This subroutine initializes the parameters used     *
C *                     throughout the program.                             *
C *                                                                         *
C ***************************************************************************
      SUBROUTINE INTPRM

      INCLUDE 'SYSVAR.COM'
C
C  BIGNUM is the psuedo-penalty when no allowable controls exist
C  RECRAT is the number of missile position recorded in each stage
C  PI is the value of pi
C  DPI is the value of pi in double precision
C
          BIGNUM=100000.0
          RECRAT = 5
          PI = ACOS(-1.0)
          DPI = DACOS(DBLE(-1.0))
C
C  These variables define the limits of the state space for x position,
C       y position, heading, and velocity.
C
          XMIN = 0.0
          XMAX = 28.0

          YMIN = 0.0
          YMAX = 28.0

          HDGMIN = -PI
          HDGMAX = PI

          VELMIN = 330.0/60.0
          VELMAX = 470.0/60.0
C
C  These variables define the size of the increment for each state
C       variable
C
          DX = 2
          DY = 2
          DHDG = PI/4
          DVEL = (VELMAX-VELMIN)/4
          DTIME = .5
```

```
C
C  These variables contain the number of quantized states in the state
C       space for each state variable
C
            WX = 15
            WY = 15
            WHDG = 8
            WVEL = 5
C
C  The bank and thrust values for each allowable control initialized
C
C       Control 1 - No turn, No acceleration
            CONTRL(1,1) = 0.0
            CONTRL(1,2) = 0.0

C       Control 2 - Right turn, No acceleration
            CONTRL(2,1) = 30.0/180.0*PI
            CONTRL(2,2) = 0.0

C       Control 3 - Left turn, No acceleration
            CONTRL(3,1) = -30.0/180.0*PI
            CONTRL(3,2) = 0.0

C       Control 4 - No turn, Max acceleration
            CONTRL(4,1) = 0.0
            CONTRL(4,2) = 1.0

C       Control 5 - No turn, Max deceleration
            CONTRL(5,1) = 0.0
            CONTRL(5,2) = -1.0

            RETURN
          END
```

```
C ***************************************************************************
C *                                                                         *
C *  SUBROUTINE:        INTDP                                               *
C *                                                                         *
C *  DESCRIPTION:       This subroutine initializes the variables used      *
C *                     primarily in the dynamic programming                *
C *                     computation of the optimal controls.                *
C *                                                                         *
C ***************************************************************************
      SUBROUTINE INTDP

      INCLUDE 'SYSVAR.COM'
      INCLUDE 'DPVAR.COM'

          DOUBLE PRECISION    NXTX,   NXTY,   NXTHDG, NXTVEL

          PRINT *,'INITIALIZING ARRAYS'
          PRINT *
C
C  Initialize the array containing the size of the increments of each
C       state for interpolation purposes.
C
          DPOS(1) = DVEL
          DPOS(2) = DHDG
          DPOS(3) = DY
          DPOS(4) = DX
C
C  For each of the controls, determine the magnitude of the change
C       for each state variable.  The value of this change is
C       independent of the stage.  Thus, this change can be
C       predetermined.
C
          DO 10 C=1,5
              DO 20 HDG=1,WHDG
                  HDGPOS = HDGMIN+(HDG-1)*DHDG
                  DO 30 VEL=1,WVEL
                      VELPOS = VELMIN+(VEL-1)*DVEL
                      NXTX = DBLE(0.0)
                      NXTY = DBLE(0.0)
                      NXTHDG = DBLE(HDGPOS)
                      NXTVEL = DBLE(VELPOS)
                      CALL NEWPOS(NXTX,NXTY,NXTHDG,NXTVEL,
     +                            CONTRL(C,1),CONTRL(C,2),DTIME)
                      XCHG(HDG,VEL,C) = SNGL(NXTX)
                      YCHG(HDG,VEL,C) = SNGL(NXTY)
                      HDGCHG(HDG,VEL,C) = SNGL(NXTHDG)-HDGPOS
   30                 CONTINUE
   20             CONTINUE
                  VELCHG(C) = SNGL(NXTVEL)-VELPOS
   10         CONTINUE
          RETURN
        END
```

```
C ***************************************************************************
C *                                                                         *
C *  SUBROUTINE:        NEWPOS                                              *
C *                                                                         *
C *  DESCRIPTION:       For a given position and controls, simulate the     *
C *                     the flight of CMMCA for the appropriate time.       *
C *                                                                         *
C ***************************************************************************
      SUBROUTINE NEWPOS(XP,YP,HP,VP,BK,TH,DT)

          DOUBLE PRECISION    XP,     YP,     HP,     VP
          REAL                BK,     TH,     DT

          REAL        TANBK,  PI,     TIMINC, STEPTM, ACCEL,  G

          INTEGER     MXSTEP
C
C  Initialize the variables
C       Max acceleration for the dynamic program is defined here.
C          currently set at 30 knots per minute
C       G --> 32.2 ft/sec**2 = 19.32 nm/min**2
C
          TANBK = TAN(BK)
          PI = ACOS(-1.0)
          TIMINC = DT/10
          ACCEL = 30.0/60.0
          G = 19.32
C
C  Determine the number of whole time increments in the time frame
C       requested and simulate flight that number of increments.
C
          MXSTEP = IFIX(DT/TIMINC)
          STEPTM = DT - MXSTEP*TIMINC
          XP = XP + (VP*DSIN(HP)+G*TANBK*DCOS(HP)*STEPTM) * STEPTM
          YP = YP + (VP*DCOS(HP)-G*TANBK*DSIN(HP)*STEPTM) * STEPTM
          HP = HP + (G*TANBK/VP) * STEPTM
          VP = VP + (ACCEL*TH) * STEPTM

          DO 10 I=1,MXSTEP
              XP = XP + (VP*DSIN(HP)+G*TANBK*DCOS(HP)*TIMINC)*TIMINC
              YP = YP + (VP*DCOS(HP)-G*TANBK*DSIN(HP)*TIMINC)*TIMINC
              HP = HP + (G*TANBK/VP) * TIMINC
              VP = VP + (ACCEL*TH) * TIMINC
10          CONTINUE
C
C  Correct the heading to lie betwee -PI and PI
C
          IF (HP.LT.-PI) HP = HP + 2*PI
          IF (HP.GT.PI)  HP = HP - 2*PI

          RETURN
      END
```

```
C ***************************************************************************
C *                                                                         *
C *  FUNCTION:          EST                                                 *
C *                                                                         *
C *  DESCRIPTION:       This function performs the linear inter-            *
C *                     polation.  The sixteen values from which the        *
C *                     interpolation is to be performed is loaded into     *
C *                     the array VAL.  The difference between the          *
C *                     the points which define the values in VAL and       *
C *                     the requested point is loaded in ADJ.  The size     *
C *                     of the increments between the established           *
C *                     values is loaded in DP.                             *
C *                                                                         *
C ***************************************************************************
      REAL FUNCTION EST(ADJ,DP,VAL)

          REAL        ADJ(4), DP(4),  VAL(16)

          VAL(1) = VAL(1) + (VAL(2)-VAL(1))/DP(1)*ADJ(1)
          VAL(3) = VAL(3) + (VAL(4)-VAL(3))/DP(1)*ADJ(1)
          VAL(5) = VAL(5) + (VAL(6)-VAL(5))/DP(1)*ADJ(1)
          VAL(7) = VAL(7) + (VAL(8)-VAL(7))/DP(1)*ADJ(1)
          VAL(9) = VAL(9) + (VAL(10)-VAL(9))/DP(1)*ADJ(1)
          VAL(11) = VAL(11) + (VAL(12)-VAL(11))/DP(1)*ADJ(1)
          VAL(13) = VAL(13) + (VAL(14)-VAL(13))/DP(1)*ADJ(1)
          VAL(15) = VAL(15) + (VAL(16)-VAL(15))/DP(1)*ADJ(1)

          VAL(1) = VAL(1) + (VAL(3)-VAL(1))/DP(2)*ADJ(2)
          VAL(5) = VAL(5) + (VAL(7)-VAL(5))/DP(2)*ADJ(2)
          VAL(9) = VAL(9) + (VAL(11)-VAL(9))/DP(2)*ADJ(2)
          VAL(13) = VAL(13) + (VAL(15)-VAL(13))/DP(2)*ADJ(2)

          VAL(1) = VAL(1) + (VAL(5)-VAL(1))/DP(3)*ADJ(3)
          VAL(9) = VAL(9) + (VAL(13)-VAL(9))/DP(3)*ADJ(3)

          EST = VAL(1) + (VAL(9)-VAL(1))/DP(4)*ADJ(4)

          RETURN
         END
```

```
C ***************************************************************************
C *                                                                         *
C *  FUNCTION:          PENLTY                                              *
C *                                                                         *
C *  DESCRIPTION:       This function determines the penalty for            *
C *                     the current position of the CMMCA and the           *
C *                     missile.                                            *
C *                                                                         *
C ***************************************************************************
      REAL FUNCTION PENLTY(XPOS,YPOS,HDG,BNK,THR,CMX,CMY,DT)

          REAL        XPOS,   YPOS,   HDG,
     +                BNK,    THR,
     +                CMX,    CMY,
     +                DT

          REAL        RANGE,  AZ,     RNGOPT, ALT,    PI
C
C  Initialize the optimal range and the altitude of CMMCA
C
          RNGOPT = 15.0
          ALT = 29000.0/6076
          PI = ACOS(-1.0)
C
C  Compute the range and azimuth of the missile as seen on the
C       radar scope.  If the CMMCA is directly above the missile,
C       assume an azimuth of PI.
C
          RANGE = SQRT((XPOS-CMX)**2+(YPOS-CMY)**2)
          IF (CMX-XPOS.NE.0.0.OR.CMY-YPOS.NE.0.0) THEN
              AZ = ATAN2(CMX-XPOS,CMY-YPOS) - HDG
              IF (AZ.LT.-PI) AZ = AZ + 2*PI
              IF (AZ.GT.PI)  AZ = AZ - 2*PI
            ELSE
              AZ = PI
            ENDIF
C
C  Compute the penalty
C
          PENLTY = (ABS(RANGE-RNGOPT) + ABS(AZ*RANGE)) * DT

          RETURN
        END
```

```
C ***************************************************************************
C *                                                                         *
C *  SUBROUTINE:        CLS                                                 *
C *                                                                         *
C *  DESCRIPTION:       Clears the screen by writting 26 blank lines.       *
C *                                                                         *
C ***************************************************************************
      SUBROUTINE CLS

        INTEGER LYNE

        DO 10 LYNE=1,26
           PRINT *
10        CONTINUE

        RETURN
      END
```

```
*************************************************************************
*                                                                       *
*  These are the common blocks that were included in the source code.   *
*                                                                       *
*************************************************************************

'SYSVAR.COM'

      COMMON/SYSVAR/  XMIN,    YMIN,    HDGMIN, VELMIN,
     +                XMAX,    YMAX,    HDGMAX, VELMAX, TMAX,
     +                DX,      DY,      DHDG,   DVEL,   DTIME,
     +                WX,      WY,      WHDG,   WVEL,
     +                RECRAT,  PI,      DPI,    CONTRL,
     +                BIGNUM

      INTEGER WX,      WY,      WHDG,   WVEL,
     +        RECRAT

      REAL    XMIN,    YMIN,    HDGMIN, VELMIN,
     +        XMAX,    YMAX,    HDGMAX, VELMAX, TMAX,
     +        DX,      DY,      DHDG,   DVEL,   DTIME,
     +        PI,      CONTRL(5,2),     BIGNUM

      DOUBLE PRECISION        DPI


'CMVAR.COM'

      COMMON/CMVAR/   XPOS,    YPOS,
     +                NEXTX,   NEXTY,
     +                ETE,     ETA,
     +                TNOW,    TRECRD,
     +                BANK,    G,
     +                SPEED,   HEADNG,
     +                XSPEED,  YSPEED,
     +                TIMINC

      REAL    ETE,     ETA,
     +        TNOW,    TRECRD,
     +        TURNRD,  NEXTRD,
     +        BANK,    G,
     +        TIMINC

      DOUBLE PRECISION        XPOS,    YPOS,
     +                        NEXTX,   NEXTY,
     +                        SPEED,   HEADNG,
     +                        XSPEED,  YSPEED
```

'DPVAR.COM'

```
      COMMON/DPVAR/   X,       Y,       HDG,     VEL,
     +                XPOS,    YPOS,    HDGPOS,  VELPOS,
     +                XCHG,    YCHG,    HDGCHG,  VELCHG,
     +                CMXPOS,  CMYPOS,  C,       COPT,
     +                DPOS,    DIFF,    OPTCNT,  PENNOW,
     +                PENLAS

      INTEGER X,       Y,       HDG,     VEL,
     +        C,       COPT,    OPTCNT(15,15,8,5)

      REAL    XPOS,    YPOS,    HDGPOS,  VELPOS,
     +        XCHG(8,5,5),      YCHG(8,5,5),
     +        HDGCHG(8,5,5),    VELCHG(5),
     +        CMXPOS,  CMYPOS,  DPOS(4),         DIFF(4),
     +        PENNOW(15,15,8,5),         PENLAS(15,15,8,5)
```

Appendix C. Turn Results

The results for three turns of the cruise missile follow. The optimal control inputs are applied and a simulation of the tracking of missile is performed. The range and azimuth of the cruise missile is presented every 0.1 minutes.

The bank control column is the number of degrees of bank the pilot should fly. A negative value signals a left turn while a positive value is a right turn.

The thrust control colum is a relative measure of the thrust the pilot should apply. The magnitude of this column is the percentage of maximum acceleration or deceleration required. A positive number is an acceleration while a negative number is a deceleration.

The desired position for the CMMCA in these runs was defined to be 15 nautical miles behind the missile (15.52 nm slant range) and zero degrees azimuth.

For each turn, the cruise missile starts out straight and level heading due north with 400 knots airspeed at sea level. At 0.6 minutes into the manuever, the missile begins its turn. The CMMCA is positioned at flight level 290 heading due north with 400 knots airpseed. The intial slant range to the cruise missile is 17.52 nautical miles.

Cruise missile turns of approximately 90 degrees, 180 degrees, and 270 degrees were simulated. CMMCA controls were computed using the dynamic programming methodology using

time increments of 0.3 minutes, 0.5 minutes, and 0.7 minutes. Figures showing the ground tracks of the CMMCA and the cruise missile are provided after the results from each computer run.

90 Degree Turn at
0.3 Minute Intervals

	OMCA PARAMETERS				CONTROLS		MISSILE POS		PERFORMANCE	
TNOW	XPOS	YPOS	HDG	VEL	BANK	THRUST	OMX	OMY	RANGE	AZIMUTH
0.00	10.000	5.000	0.	400.	0.0	-1.00	10.000	20.000	15.52	0.00
0.10	10.000	5.664	0.	397.	0.0	-1.00	10.000	20.667	15.53	0.00
0.20	10.000	6.324	0.	394.	0.0	-1.00	10.000	21.333	15.53	0.00
0.30	10.000	6.978	0.	391.	0.0	-1.00	10.000	22.000	15.55	0.00
0.40	10.000	7.628	0.	388.	0.0	-1.00	10.000	22.667	15.56	0.00
0.50	10.000	8.272	0.	385.	0.0	-1.00	10.000	23.333	15.58	0.00
0.60	10.000	8.911	0.	382.	0.0	-1.00	10.000	24.000	15.61	0.00
0.70	10.000	9.546	0.	379.	0.0	-1.00	10.061	24.663	15.64	0.23
0.80	10.000	10.175	0.	376.	0.0	-1.00	10.232	25.307	15.65	0.88
0.90	10.000	10.800	0.	373.	0.0	-1.00	10.508	25.913	15.64	1.92
1.00	10.000	11.419	0.	370.	0.0	-1.00	10.880	26.465	15.59	3.35
1.10	10.000	12.034	0.	367.	0.0	-1.00	11.340	26.947	15.50	5.13
1.20	10.000	12.643	0.	364.	9.6	-0.66	11.873	27.346	15.35	7.26
1.30	10.018	13.248	3.	362.	9.6	-0.66	12.465	27.650	15.15	6.53
1.40	10.069	13.848	6.	360.	9.6	-0.66	13.100	27.852	14.88	5.98
1.50	10.152	14.440	9.	358.	23.8	-0.16	13.759	27.945	14.54	5.58
1.60	10.295	15.019	18.	358.	23.8	-0.16	14.425	27.927	14.13	0.16
1.70	10.518	15.570	26.	357.	23.8	-0.16	15.089	27.865	13.71	-5.40
1.80	10.818	16.084	34.	357.	26.9	0.00	15.752	27.803	13.33	-11.18
1.90	11.194	16.544	43.	357.	26.9	0.00	16.416	27.741	12.99	-18.47
2.00	11.640	16.936	53.	357.	26.9	0.00	17.080	27.679	12.69	-26.08
2.10	12.144	17.249	62.	357.	16.7	-0.27	17.744	27.616	12.44	-34.03
2.20	12.684	17.495	68.	356.	16.7	-0.27	18.407	27.554	12.24	-38.37
2.30	13.245	17.687	74.	355.	16.7	-0.27	19.071	27.492	12.09	-42.91
2.40	13.820	17.823	79.	354.	4.5	-0.49	19.735	27.430	11.97	-47.64
2.50	14.400	17.924	81.	353.	4.5	-0.49	20.399	27.368	11.88	-48.32
2.60	14.980	18.011	82.	351.	4.5	-0.49	21.062	27.306	11.81	-49.03
2.70	15.560	18.081	84.	350.	-6.6	-0.49	21.726	27.243	11.75	-49.78
2.80	16.137	18.157	82.	348.	-6.6	-0.49	22.390	27.181	11.68	-46.80
2.90	16.708	18.255	79.	347.	-6.6	-0.49	23.054	27.119	11.61	-43.70
3.00	17.273	18.374	77.	345.	-15.8	-0.16	23.717	27.057	11.53	-40.49
3.10	17.826	18.532	72.	345.	-15.8	-0.16	24.381	26.995	11.43	-33.88
3.20	18.361	18.741	66.	344.	-15.8	-0.16	25.045	26.933	11.30	-26.98
3.30	18.872	18.999	61.	344.	-10.5	-0.26	25.709	26.870	11.17	-19.77
3.40	19.362	19.296	57.	343.	-10.5	-0.26	26.372	26.808	11.03	-14.14
3.50	19.831	19.622	54.	342.	-10.5	-0.26	27.036	26.746	10.89	-8.25
3.60	20.278	19.976	50.	342.	-3.3	-0.29	27.700	26.684	10.77	-2.08
3.70	20.709	20.347	49.	341.	-3.3	-0.29	28.364	26.622	10.68	1.81
3.80	21.132	20.724	48.	340.	-3.3	-0.29	29.027	26.559	10.60	5.82
3.90	21.547	21.110	47.	339.	2.6	-0.22	29.691	26.497	10.55	9.93
4.00	21.960	21.494	47.	338.	2.6	-0.22	30.355	26.435	10.53	12.05

4.10	22.378	21.871	48.	338.	2.6	-0.22	31.019	26.373	10.53	14.13
4.20	22.801	22.241	49.	337.	26.2	-0.01	31.683	26.311	10.56	16.14
4.30	23.259	22.567	59.	337.	26.2	-0.01	32.346	25.249	10.59	9.01
4.40	23.764	22.810	69.	337.	26.2	-0.01	33.010	26.186	10.62	1.32
4.50	24.303	22.963	78.	337.			33.674	26.124	10.67	-6.94

AVERAGE TIME WEIGHTED DEVIATIONS FROM DESIRED POSITION = 5.782142

90 Degree Turn
0.3 Min Increment

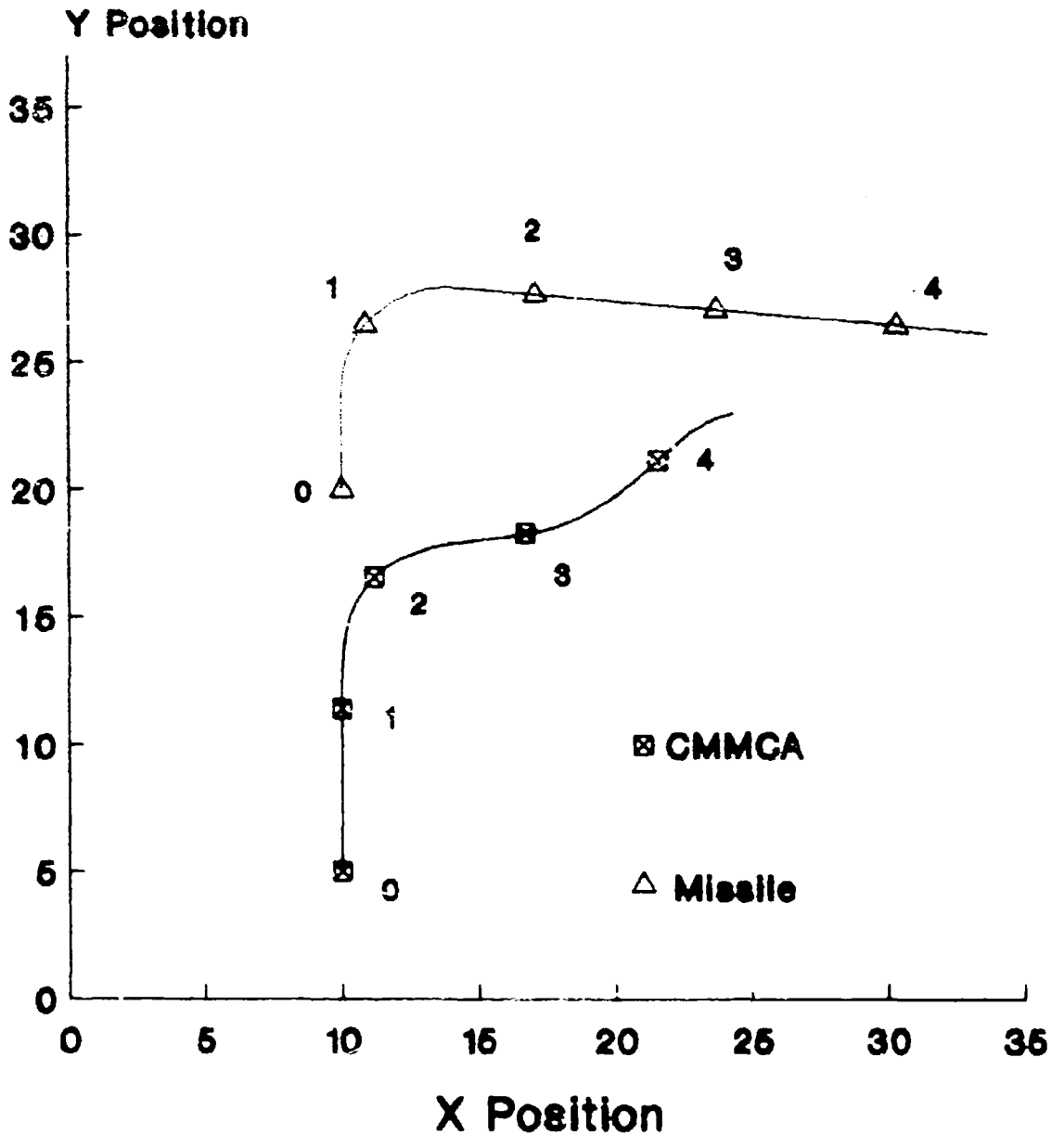

Figure 1. 90 Degree Turn at 0.3 Minute Increments

180 Degree Turn at
0.3 Minute Intervals

	OMCA PARAMETERS				CONTROLS		MISSILE POS		PERFORMANCE	
TNOW	XPOS	YPOS	HDG	VEL	BANK	THRUST	OMX	OMY	RANGE	AZIMUTH
0.00	10.000	5.000	0.	400.	0.0	-1.00	10.000	20.000	15.52	0.00
0.10	10.000	5.664	0.	397.	0.0	-1.00	10.000	20.667	15.53	0.00
0.20	10.000	6.324	0.	394.	0.0	-1.00	10.000	21.333	15.53	0.00
0.30	10.000	6.978	0.	391.	0.0	-1.00	10.000	22.000	15.55	0.00
0.40	10.000	7.628	0.	388.	0.0	-1.00	10.000	22.667	15.56	0.00
0.50	10.000	8.272	0.	385.	0.0	-1.00	10.000	23.333	15.58	0.00
0.60	10.000	8.911	0.	382.	0.0	-1.00	10.000	24.000	15.61	0.00
0.70	10.000	9.546	0.	379.	0.0	-1.00	10.061	24.663	15.64	0.25
0.80	10.000	10.175	0.	376.	0.0	-1.00	10.232	25.307	15.65	0.88
0.90	10.000	10.800	0.	373.	0.0	-1.00	10.508	25.913	15.64	1.92
1.00	10.000	11.419	0.	370.	0.0	-1.00	10.880	26.465	15.59	3.35
1.10	10.000	12.034	0.	367.	0.0	-1.00	11.340	26.947	15.50	5.13
1.20	10.000	12.643	0.	364.	0.0	-0.97	11.873	27.346	15.35	7.26
1.30	10.000	13.247	0.	361.	0.0	-0.97	12.465	27.650	15.15	9.71
1.40	10.000	13.847	0.	358.	0.0	-0.97	13.100	27.852	14.89	12.48
1.50	10.000	14.442	0.	355.	13.1	-0.56	13.759	27.945	14.58	15.56
1.60	10.025	15.032	4.	354.	13.1	-0.56	14.425	27.927	14.20	14.47
1.70	10.094	15.616	9.	352.	13.1	-0.56	15.078	27.798	13.76	13.49
1.80	10.208	16.190	13.	350.	22.9	-0.14	15.701	27.563	13.25	12.60
1.90	10.384	16.746	21.	350.	22.9	-0.14	16.276	27.226	12.67	8.16
2.00	10.635	17.271	29.	349.	22.9	-0.14	16.787	26.799	12.03	3.65
2.10	10.957	17.755	37.	349.	28.4	-0.03	17.219	26.293	11.32	-0.97
2.20	11.353	18.181	48.	349.	28.4	-0.03	17.561	25.722	10.56	-8.06
2.30	11.817	18.529	58.	349.	28.4	-0.03	17.804	25.102	9.75	-15.50
2.40	12.337	18.788	68.	349.	19.9	-0.14	17.940	24.450	8.91	-23.43
2.50	12.889	18.968	75.	348.	19.9	-0.14	17.965	23.784	8.06	-28.51
2.60	13.458	19.080	82.	348.	19.9	-0.14	17.909	23.120	7.22	-34.14
2.70	14.035	19.124	89.	347.	30.0	0.00	17.849	22.456	6.45	-39.97
2.80	14.611	19.074	100.	347.	30.0	0.00	17.789	21.792	5.79	-50.40
2.90	15.168	18.916	111.	347.	30.0	0.00	17.729	21.128	5.24	-61.72
3.00	15.683	18.654	122.	347.	17.6	-0.06	17.669	20.464	4.82	-74.29
3.10	16.155	18.319	128.	347.	17.6	-0.06	17.609	19.800	4.51	-83.56
3.20	16.590	17.937	134.	347.	17.6	-0.06	17.549	19.136	4.28	-95.45
3.30	16.981	17.511	140.	347.	20.4	-0.03	17.489	18.472	4.15	-112.34
3.40	17.320	17.043	147.	347.	20.4	-0.03	17.429	17.808	4.07	-139.18
3.50	17.598	16.537	154.	347.	20.4	-0.03	17.369	17.144	4.05	-175.07
3.60	17.811	16.000	162.	347.	26.9	-0.03	17.309	16.480	4.06	152.19
3.70	17.943	15.438	171.	347.	26.9	-0.03	17.249	15.816	4.08	127.39
3.80	17.978	14.862	181.	347.	26.9	-0.03	17.189	15.153	4.09	109.30
3.90	17.915	14.289	191.	346.	20.8	-0.11	17.129	14.489	4.08	93.65
4.00	17.769	13.730	198.	346.	20.8	-0.11	17.069	13.825	4.06	79.75

4.10	17.554	13.196	205.	346.	20.8	-0.11	17.009	13.161	4.04	61.11
4.20	17.273	12.693	213.	345.	-4.5	-0.22	16.949	12.497	4.02	26.30
4.30	16.971	12.204	211.	345.	-4.5	-0.22	16.889	11.833	4.02	-18.41
4.40	16.683	11.707	209.	344.	-4.5	-0.22	16.829	11.169	4.04	-44.58
4.50	16.409	11.204	208.	344.	-14.7	-0.17	16.769	10.505	4.09	-55.15
4.60	16.166	10.686	203.	343.	-14.7	-0.17	16.709	9.841	4.12	-35.54
4.70	15.970	10.149	198.	343.	-14.7	-0.17	16.649	9.177	4.17	-52.65
4.80	15.823	9.598	193.	342.	-10.4	-0.22	16.589	8.513	4.21	-47.86
4.90	15.717	9.039	189.	341.	-10.4	-0.22	16.529	7.849	4.25	-43.41
5.00	15.646	8.475	186.	341.	-10.4	-0.22	16.469	7.185	4.28	-38.08
5.10	15.611	7.909	182.	340.	-6.9	-0.22	16.409	6.521	4.31	-31.89
5.20	15.604	7.343	180.	339.	-6.9	-0.22	16.349	5.857	4.33	-26.25
5.30	15.620	6.778	177.	339.	-6.9	-0.22	16.289	5.193	4.35	-20.11
5.40	15.660	6.215	175.	338.	-1.6	-0.17	16.229	4.529	4.38	-13.49
5.50	15.714	5.655	174.	338.	-1.6	-0.17	16.169	3.865	4.41	-8.58
5.60	15.772	5.096	174.	337.	-1.6	-0.17	16.109	3.201	4.44	-3.83
5.70	15.836	4.538	173.	336.			16.049	2.537	4.48	0.74

AVERAGE TIME WEIGHTED DEVIATIONS FROM DESIRED POSITION = 9.950198

180 Degree Turn
0.3 Min Increment

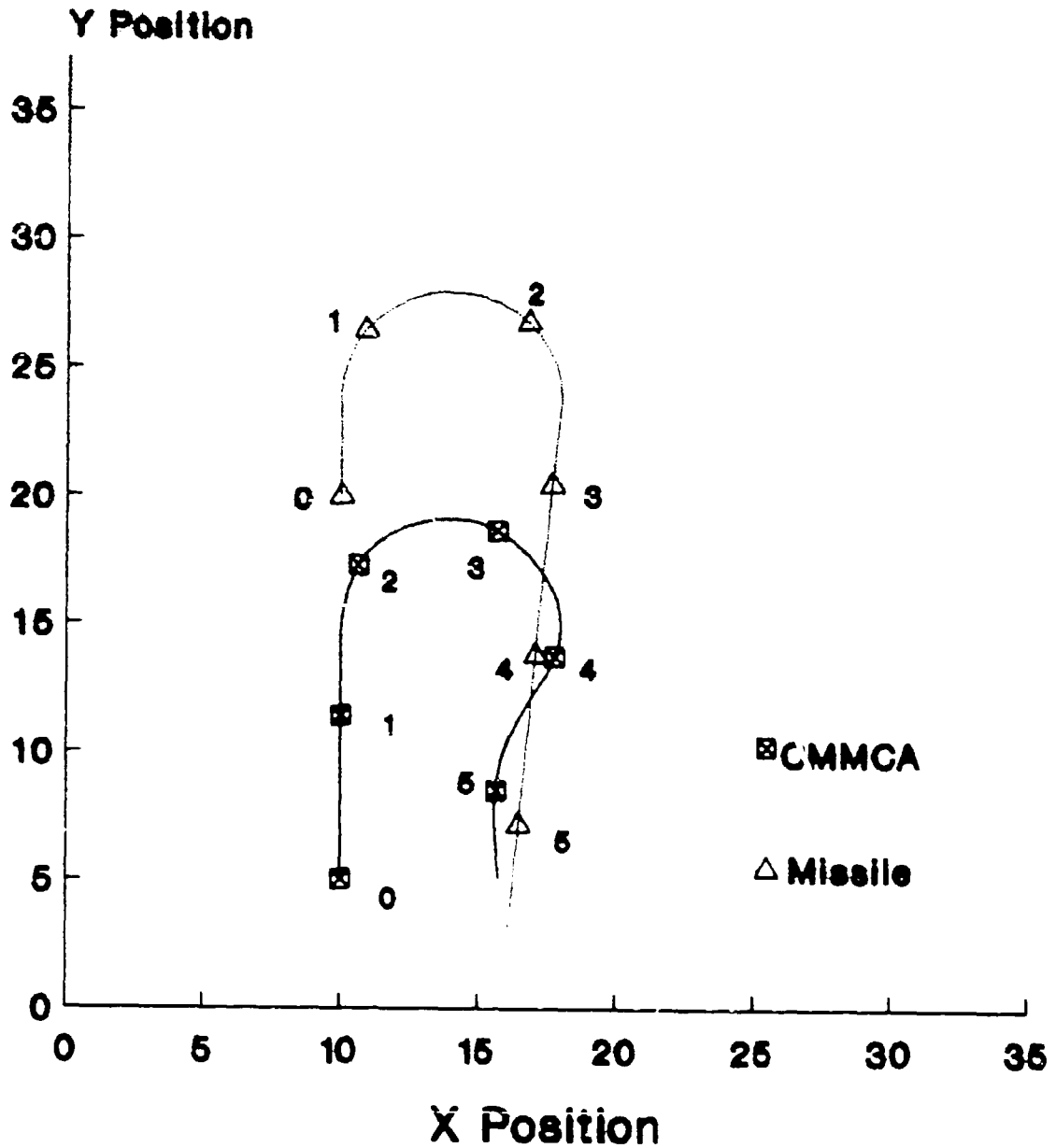

Figure 2. 180 Degree Turn at 0.3 Minute Increments

270 Degree Turn at
0.3 Minute Intervals

	OMCA PARAMETERS				CONTROLS		MISSILE POS		PERFORMANCE	
TNOW	XPOS	YPOS	HDG	VEL	BANK	THRUST	OMX	OMY	RANGE	AZIMUTH
0.00	10.000	5.000	0.	400.	0.0	-1.00	10.000	20.000	15.52	0.00
0.10	10.000	5.664	0.	397.	0.0	-1.00	10.000	20.667	15.53	0.00
0.20	10.000	6.324	0.	394.	0.0	-1.00	10.000	21.333	15.53	0.00
0.30	10.000	6.978	0.	391.	0.0	-1.00	10.000	22.000	15.55	0.00
0.40	10.000	7.628	0.	388.	0.0	-1.00	10.000	22.667	15.56	0.00
0.50	10.000	8.272	0.	385.	0.0	-1.00	10.000	23.333	15.58	0.00
0.60	10.000	8.911	0.	382.	0.0	-1.00	10.000	24.000	15.61	0.00
0.70	10.000	9.546	0.	379.	0.0	-1.00	9.939	24.663	15.64	-0.23
0.80	10.000	10.175	0.	376.	0.0	-1.00	9.768	25.307	15.65	-0.88
0.90	10.000	10.800	0.	373.	0.0	-1.00	9.492	25.913	15.64	-1.92
1.00	10.000	11.419	0.	370.	0.0	-1.00	9.120	26.465	15.59	-3.35
1.10	10.000	12.034	0.	367.	0.0	-1.00	8.660	26.947	15.50	-5.13
1.20	10.000	12.643	0.	364.	0.3	-0.97	8.127	27.346	15.35	-7.26
1.30	10.001	13.247	0.	361.	0.3	-0.97	7.535	27.650	15.15	-9.80
1.40	10.002	13.847	0.	358.	0.3	-0.97	6.900	27.852	14.89	-12.66
1.50	10.004	14.442	0.	355.	13.0	-0.56	6.241	27.945	14.58	-15.84
1.60	10.032	15.032	5.	354.	13.0	-0.56	5.575	27.927	14.22	-23.66
1.70	10.103	15.615	9.	352.	13.0	-0.56	4.922	27.798	13.83	-31.99
1.80	10.218	16.189	13.	350.	21.6	-0.17	4.299	27.563	13.43	-40.02
1.90	10.393	16.745	21.	350.	21.6	-0.17	3.724	27.226	13.05	-53.30
2.00	10.639	17.273	28.	349.	21.6	-0.17	3.213	26.799	12.72	-66.28
2.10	10.951	17.763	36.	349.	16.7	-0.25	2.781	26.293	12.47	-79.64
2.20	11.316	18.214	42.	348.	16.7	-0.25	2.439	25.722	12.30	-91.39
2.30	11.724	18.625	47.	347.	16.7	-0.25	2.196	25.102	12.20	-103.15
2.40	12.170	18.993	53.	346.	24.9	-0.08	2.060	24.450	12.16	-114.75
2.50	12.659	19.298	62.	346.	24.9	-0.08	2.035	23.784	12.21	-129.14
2.60	13.190	19.524	71.	346.	24.9	-0.08	2.121	23.124	12.3[illegible]	-142.94
2.70	13.748	19.664	80.	346.	25.5	-0.09	2.316	22.487	12.44	-156.02
2.80	14.321	19.715	89.	345.	25.5	-0.09	2.614	21.892	12.56	-168.54
2.90	14.895	19.674	98.	345.	25.5	-0.09	3.007	21.354	12.65	179.78
3.00	15.454	19.541	107.	345.	17.9	-0.18	3.484	20.890	12.69	168.97
3.10	15.990	19.337	114.	344.	17.9	-0.18	4.032	20.511	12.66	161.93
3.20	16.501	19.076	120.	344.	17.9	-0.18	4.635	20.229	12.57	155.65
3.30	15.979	18.761	126.	343.	17.7	-0.16	5.277	20.051	12.43	150.16
3.40	17.419	18.398	132.	343.	17.7	-0.16	5.939	19.983	12.26	145.55
3.50	17.818	17.989	138.	342.	17.7	-0.16	6.606	20.000	12.07	141.67
3.60	18.170	17.541	145.	342.	23.5	-0.07	7.272	20.024	11.87	138.14
3.70	18.460	17.051	153.	342.	23.5	-0.07	7.938	20.047	11.65	132.74
3.80	18.674	16.524	162.	341.	23.5	-0.07	8.604	20.071	11.40	127.77
3.90	18.809	15.972	170.	341.	16.3	-0.15	9.271	20.094	11.13	123.26
4.00	18.876	15.408	176.	341.	16.3	-0.15	9.937	20.118	10.87	122.00

4.10	18.887	14.841	181.	340.	16.3	-0.15	10.603	20.141	10.62	121.14
4.20	18.841	14.276	187.	340.	22.9	-0.06	11.269	20.165	10.39	120.70
4.30	18.726	13.722	195.	340.	22.9	-0.06	11.936	20.188	10.19	118.16
4.40	18.533	13.190	204.	339.	22.9	-0.06	12.602	20.212	10.02	116.10
4.50	18.265	12.692	212.	339.	28.3	-0.06	13.268	20.235	9.89	114.48
4.60	17.919	12.246	223.	339.	28.3	-0.06	13.934	20.258	9.80	111.02
4.70	17.498	11.871	233.	339.	28.3	-0.06	14.601	20.282	9.75	107.91
4.80	17.015	11.580	244.	339.	28.8	-0.04	15.267	20.305	9.76	105.04
4.90	16.486	11.383	254.	339.	28.8	-0.04	15.933	20.329	9.82	102.05
5.00	15.931	11.288	265.	339.	28.8	-0.04	16.599	20.352	9.93	99.03
5.10	15.368	11.299	276.	338.	30.0	0.00	17.266	20.376	10.10	95.84
5.20	14.817	11.418	287.	338.	30.0	0.00	17.932	20.399	10.31	91.83
5.30	14.301	11.643	299.	338.	30.0	0.00	18.598	20.423	10.56	87.45
5.40	13.839	11.965	310.	338.	30.0	0.00	19.264	20.446	10.83	82.65
5.50	13.449	12.371	321.	338.	30.0	0.00	19.931	20.470	11.12	77.38
5.60	13.146	12.847	333.	338.	30.0	0.00	20.597	20.493	11.40	71.63
5.70	12.943	13.372	344.	338.	30.0	0.00	21.263	20.516	11.67	65.39
5.80	12.847	13.927	355.	338.	30.0	0.00	21.929	20.540	11.93	58.65
5.90	12.862	14.490	7.	338.	30.0	0.00	22.596	20.563	12.15	51.42
6.00	12.987	15.039	18.	338.	18.0	-0.10	23.262	20.587	12.34	43.68
6.10	13.194	15.564	24.	338.	18.0	-0.10	23.928	20.610	12.52	40.48
6.20	13.457	16.061	31.	338.	18.0	-0.10	24.594	20.634	12.69	36.94
6.30	13.773	16.526	37.	338.	13.8	-0.12	25.261	20.657	12.85	33.08
6.40	14.133	16.958	42.	337.	13.8	-0.12	25.927	20.681	13.00	30.50
6.50	14.528	17.358	47.	337.	13.8	-0.12	26.593	20.704	13.14	27.66
6.60	14.955	17.722	52.	337.	25.2	-0.01	27.259	20.728	13.28	24.59
6.70	15.423	18.029	61.	336.	25.2	-0.01	27.926	20.751	13.41	16.72
6.80	15.936	18.256	70.	336.	25.2	-0.01	28.592	20.774	13.51	8.44
6.90	16.478	18.397	80.	336.	6.9	-0.14	29.258	20.798	13.61	-0.25
7.00	17.031	18.485	82.	336.	6.9	-0.14	29.924	20.821	13.70	-2.28
7.10	17.587	18.550	84.	336.	6.9	-0.14	30.591	20.845	13.80	-4.42
7.20	18.144	18.591	87.	335.	2.1	-0.14	31.257	20.868	13.90	-6.67
7.30	18.702	18.618	88.	335.	2.1	-0.14	31.923	20.892	14.00	-7.31
7.40	19.259	18.638	88.	334.	2.1	-0.14	32.589	20.915	14.10	-7.98
7.50	19.816	18.651	89.	334.			33.256	20.939	14.21	-8.68

AVERAGE TIME WEIGHTED DEVIATIONS FROM DESIRED POSITION = 16.31599

270 Degree Turn
0.3 Min Increment

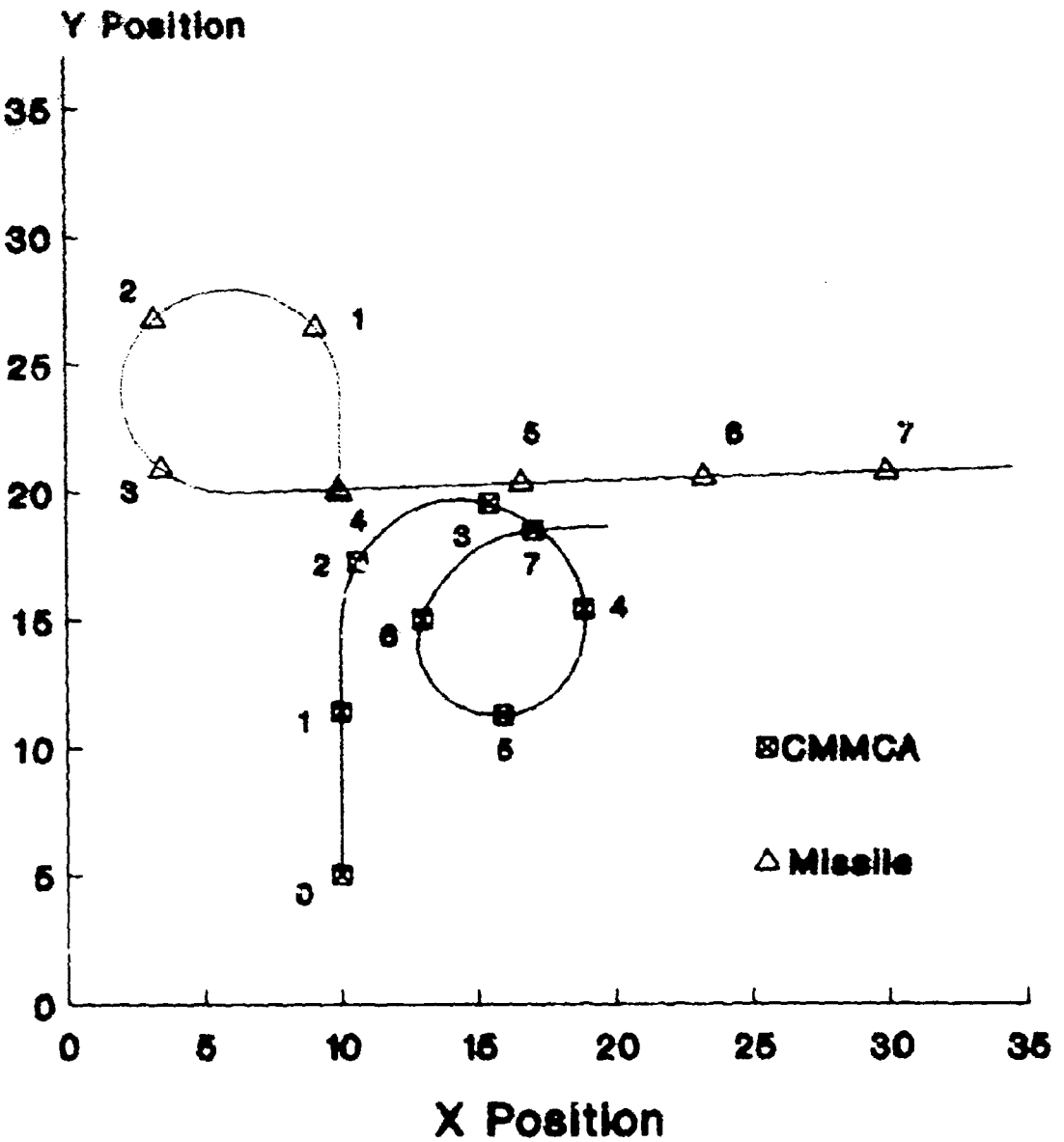

Figure 3. 270 Degree Turn at 0.3 Minute Increments

90 Degree Turn at
0.5 Minute Intervals

	OMCA PARAMETERS				CONTROLS		MISSILE POS		PERFORMANCE	
TMIN	XPOS	YPOS	HDG	VEL	BANK	THRUST	OMX	OMY	RANGE	AZIMUTH
0.00	10.000	5.000	0.	400.	0.0	-1.00	10.000	20.000	15.52	0.00
0.10	10.000	5.664	0.	397.	0.0	-1.00	10.000	20.667	15.53	0.00
0.20	10.000	6.324	0.	394.	0.0	-1.00	10.000	21.333	15.53	0.00
0.30	10.000	6.978	0.	391.	0.0	-1.00	10.000	22.000	15.55	0.00
0.40	10.000	7.628	0.	388.	0.0	-1.00	10.000	22.667	15.56	0.00
0.50	10.000	8.272	0.	385.	0.0	-1.00	10.000	23.333	15.58	0.00
0.60	10.000	8.911	0.	382.	0.0	-1.00	10.000	24.000	15.61	0.00
0.70	10.000	9.546	0.	[illegible]	0.0	-1.00	10.061	24.663	15.64	0.23
0.80	10.000	10.175	0.	[illegible]	0.0	-1.00	10.232	25.307	15.65	0.88
0.90	10.000	10.800	0.	373.	0.0	-1.00	10.508	25.913	15.64	1.92
1.00	10.000	11.419	0.	370.	0.0	-1.00	10.880	26.465	15.59	3.35
1.10	10.000	12.034	0.	367.	0.0	-1.00	11.340	26.947	15.50	5.13
1.20	10.000	12.643	0.	364.	0.0	-1.00	11.873	27.346	15.35	7.26
1.30	10.000	13.247	0.	361.	0.0	-1.00	12.465	27.650	15.15	9.71
1.40	10.000	13.847	0.	358.	0.0	-1.00	13.100	27.852	14.89	12.48
1.50	10.000	14.441	0.	355.	0.0	0.29	13.759	27.945	14.58	15.56
1.60	10.000	15.034	0.	356.	0.0	0.29	14.425	27.927	14.21	18.94
1.70	10.000	15.627	0.	357.	0.0	0.29	15.089	27.865	13.84	22.58
1.80	10.000	16.222	0.	358.	0.0	0.29	15.752	27.803	13.54	26.41
1.90	10.000	16.819	0.	358.	0.0	0.29	16.416	27.741	13.28	30.43
2.00	10.000	17.417	0.	359.	30.0	0.00	17.080	27.679	13.09	34.60
2.10	10.061	18.012	11.	359.	30.0	0.00	17.744	27.616	12.93	27.98
2.20	10.231	18.585	21.	359.	30.0	0.00	18.407	27.554	12.78	21.01
2.30	10.505	19.117	32.	359.	30.0	0.00	19.071	27.492	12.63	13.63
2.40	10.872	19.589	43.	359.	30.0	0.00	19.735	27.430	12.49	5.81
2.50	11.321	19.985	53.	359.	-5.6	0.13	20.399	27.368	12.37	-2.48
2.60	11.795	20.351	52.	360.	-5.6	0.13	21.062	27.306	12.26	1.55
2.70	12.258	20.732	50.	360.	-5.6	0.13	21.726	27.243	12.17	5.72
2.80	12.710	21.127	48.	360.	-5.6	0.13	22.390	27.181	12.10	10.02
2.90	13.149	21.538	46.	361.	-5.6	0.13	23.054	27.119	12.05	14.44
3.00	13.576	21.962	44.	361.	0.4	0.99	23.717	27.057	12.03	18.96
3.10	13.999	22.393	45.	364.	0.4	0.99	24.381	26.995	12.04	21.60
3.20	14.427	22.827	45.	367.	0.4	0.99	25.045	26.933	12.07	24.23
3.30	14.859	23.264	45.	370.	0.4	0.99	25.709	26.870	12.11	26.85
3.40	15.296	23.703	45.	373.	0.4	0.99	26.372	26.808	12.18	29.44
3.50	15.737	24.145	45.	376.	30.0	0.00	27.036	26.746	12.27	32.00
3.60	16.221	24.542	55.	376.	30.0	0.00	27.700	26.684	12.34	24.21
3.70	16.767	24.847	65.	376.	30.0	0.00	28.364	26.622	12.39	15.89
3.80	17.359	25.051	76.	376.	30.0	0.00	29.027	26.559	12.43	7.04
3.90	17.978	25.146	86.	376.	30.0	0.00	29.691	26.497	12.45	-2.36
4.00	18.604	25.131	96.	376.	-4.0	-0.87	30.355	26.435	12.48	-12.30

4.10	19.226	25.074	95.	373.	-4.0	-0.87	31.019	26.373	12.52	-11.02
4.20	19.845	25.030	93.	371.	-4.0	-0.87	31.683	26.311	12.56	-9.67
4.30	20.461	25.000	92.	368.	-4.0	-0.87	32.346	26.249	12.60	-8.24
4.40	21.072	24.983	91.	366.	-4.0	-0.87	33.010	26.186	12.65	-6.74
4.50	21.680	24.980	90.	363.			33.674	26.124	12.70	-5.17

AVERAGE TIME WEIGHTED DEVIATIONS FROM DESIRED POSITION = 4.529917

90 Degree Turn
0.5 Min Increment

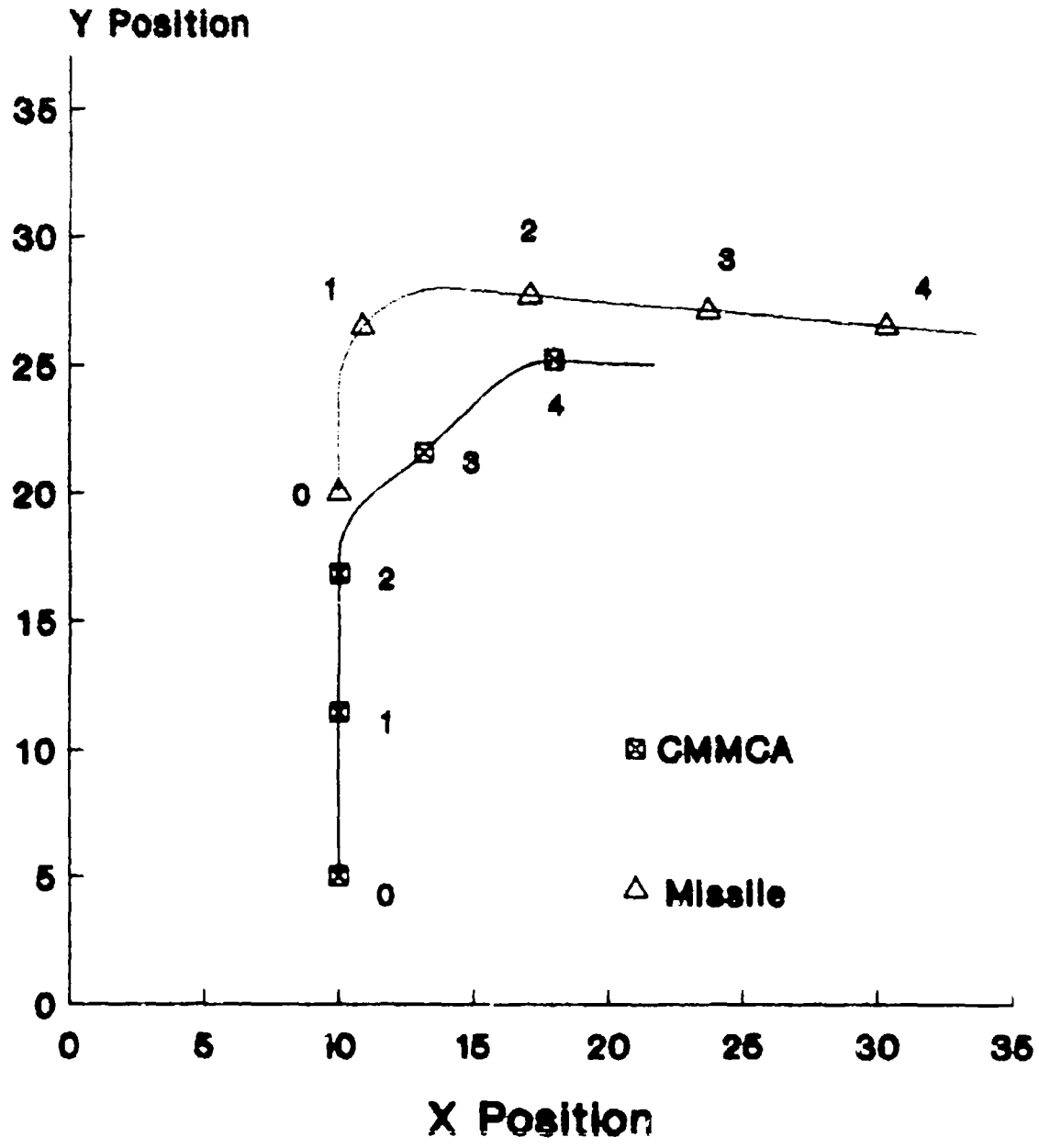

Figure 4. 90 Degree Turn at 0.5 Minute Increments

180 Degree Turn at
0.5 Minute Intervals

	OMCA PARAMETERS				CONTROLS		MISSILE POS		PERFORMANCE	
TNOW	XPOS	YPOS	HDG	VEL	BANK	THRUST	OMX	OMY	RANGE	AZIMUTH
0.00	10.000	5.000	0.	400.	0.0	-1.00	10.000	20.000	15.52	0.00
0.10	10.000	5.664	0.	397.	0.0	-1.00	10.000	20.667	15.53	0.00
0.20	10.000	6.324	0.	394.	0.0	-1.00	10.000	21.333	15.53	0.00
0.30	10.000	6.978	0.	391.	0.0	-1.00	10.000	22.000	15.55	0.00
0.40	10.000	7.628	0.	388.	0.0	-1.00	10.000	22.667	15.56	0.00
0.50	10.000	8.272	0.	385.	0.0	-1.00	10.000	23.333	15.58	0.00
0.60	10.000	8.911	0.	382.	0.0	-1.00	10.000	24.000	15.61	0.00
0.70	10.000	9.546	0.	379.	0.0	-1.00	10.061	24.663	15.64	0.23
0.80	10.000	10.175	0.	376.	0.0	-1.00	10.232	25.307	15.65	0.88
0.90	10.000	10.800	0.	373.	0.0	-1.00	10.508	25.913	15.64	1.92
1.00	10.000	11.419	0.	370.	0.0	-1.00	10.880	26.465	15.59	3.35
1.10	10.000	12.034	0.	367.	0.0	-1.00	11.340	26.947	15.50	5.13
1.20	10.000	12.643	0.	364.	0.0	-1.00	11.873	27.346	15.35	7.26
1.30	10.000	13.247	0.	361.	0.0	-1.00	12.465	27.650	15.15	9.71
1.40	10.000	13.847	0.	358.	0.0	-1.00	13.100	27.852	14.89	12.48
1.50	10.000	14.441	0.	355.	0.0	-0.71	13.759	27.945	14.58	15.56
1.60	10.000	15.031	0.	353.	0.0	-0.71	14.425	27.927	14.21	18.94
1.70	10.000	15.618	0.	351.	0.0	-0.71	15.078	27.798	13.79	22.63
1.80	10.000	16.201	0.	349.	0.0	-0.71	15.701	27.563	13.33	26.65
1.90	10.000	16.780	0.	346.	0.0	-0.71	16.276	27.226	12.83	31.00
2.00	10.000	17.356	0.	344.	0.0	-0.41	16.787	26.799	12.30	35.70
2.10	10.000	17.929	0.	343.	0.0	-0.41	17.219	26.293	11.75	40.80
2.20	10.000	18.500	0.	342.	0.0	-0.41	17.561	25.722	11.20	46.31
2.30	10.000	19.068	0.	341.	0.0	-0.41	17.804	25.102	10.64	52.29
2.40	10.000	19.635	0.	339.	0.0	-0.41	17.940	24.450	10.11	58.77
2.50	10.000	20.200	0.	338.	30.0	0.00	17.965	23.784	9.61	65.77
2.60	10.061	20.759	11.	338.	30.0	0.00	17.909	23.120	9.12	61.92
2.70	10.231	21.296	23.	338.	30.0	0.00	17.849	22.456	8.68	58.66
2.80	10.503	21.789	34.	338.	30.0	0.00	17.789	21.792	8.31	55.95
2.90	10.867	22.218	45.	338.	30.0	0.00	17.729	21.128	8.02	53.67
3.00	11.308	22.568	57.	338.	22.5	-0.05	17.669	20.464	7.80	51.60
3.10	11.801	22.839	65.	338.	22.5	-0.05	17.609	19.800	7.68	52.77
3.20	12.328	23.038	73.	338.	22.5	-0.05	17.549	19.136	7.65	53.76
3.30	12.877	23.160	81.	338.	22.5	-0.05	17.489	18.472	7.70	54.30
3.40	13.438	23.202	89.	338.	22.5	-0.05	17.429	17.808	7.81	54.17
3.50	13.999	23.165	97.	337.	25.0	-0.04	17.369	17.144	7.97	53.26
3.60	14.547	23.043	107.	337.	25.0	-0.04	17.309	16.480	8.17	50.50
3.70	15.069	22.835	116.	337.	25.0	-0.04	17.249	15.816	8.37	46.88
3.80	15.550	22.546	125.	337.	25.0	-0.04	17.189	15.153	8.56	42.46
3.90	15.980	22.185	134.	337.	25.0	-0.04	17.129	14.489	8.75	37.27
4.00	16.346	21.759	143.	337.	24.4	-0.04	17.069	13.825	8.92	31.36

4.10	16.640	21.282	152.	337.	24.4	-0.04	17.009	13.161	9.06	25.04
4.20	16.857	20.765	161.	337.	24.4	-0.04	16.949	12.497	9.19	18.06
4.30	16.990	20.220	170.	337.	24.4	-0.04	16.889	11.833	9.29	10.45
4.40	17.037	19.662	179.	336.	24.4	-0.04	15.829	11.169	9.39	2.22
4.50	16.997	19.103	188.	336.	-5.4	-0.15	16.769	10.505	9.49	-6.62
4.60	16.928	18.547	186.	336.	-3.4	-0.15	16.709	9.841	9.58	-4.82
4.70	16.877	17.990	184.	335.	-5.4	-0.15	16.649	9.177	9.68	-2.90
4.80	16.844	17.432	183.	335.	-5.4	-0.15	16.589	8.513	9.78	-0.86
4.90	16.830	16.875	181.	335.	-5.4	-0.15	16.529	7.849	9.88	1.30
5.00	16.834	16.317	179.	334.	0.8	-0.11	16.469	7.185	9.98	3.56
5.10	16.845	15.761	179.	334.	0.8	-0.11	16.409	6.521	10.08	3.68
5.20	16.853	15.205	179.	333.	0.8	-0.11	16.349	5.857	10.18	3.77
5.30	16.858	14.650	180.	333.	0.8	-0.11	16.289	5.193	10.28	3.83
5.40	16.860	14.095	180.	333.	0.8	-0.11	16.229	4.529	10.39	3.87
5.50	16.859	13.540	180.	332.			16.169	3.865	10.49	3.88

AVERAGE TIME WEIGHTED DEVIATIONS FROM DESIRED POSITION = 7.939511

180 Degree Turn
0.5 Min Increment

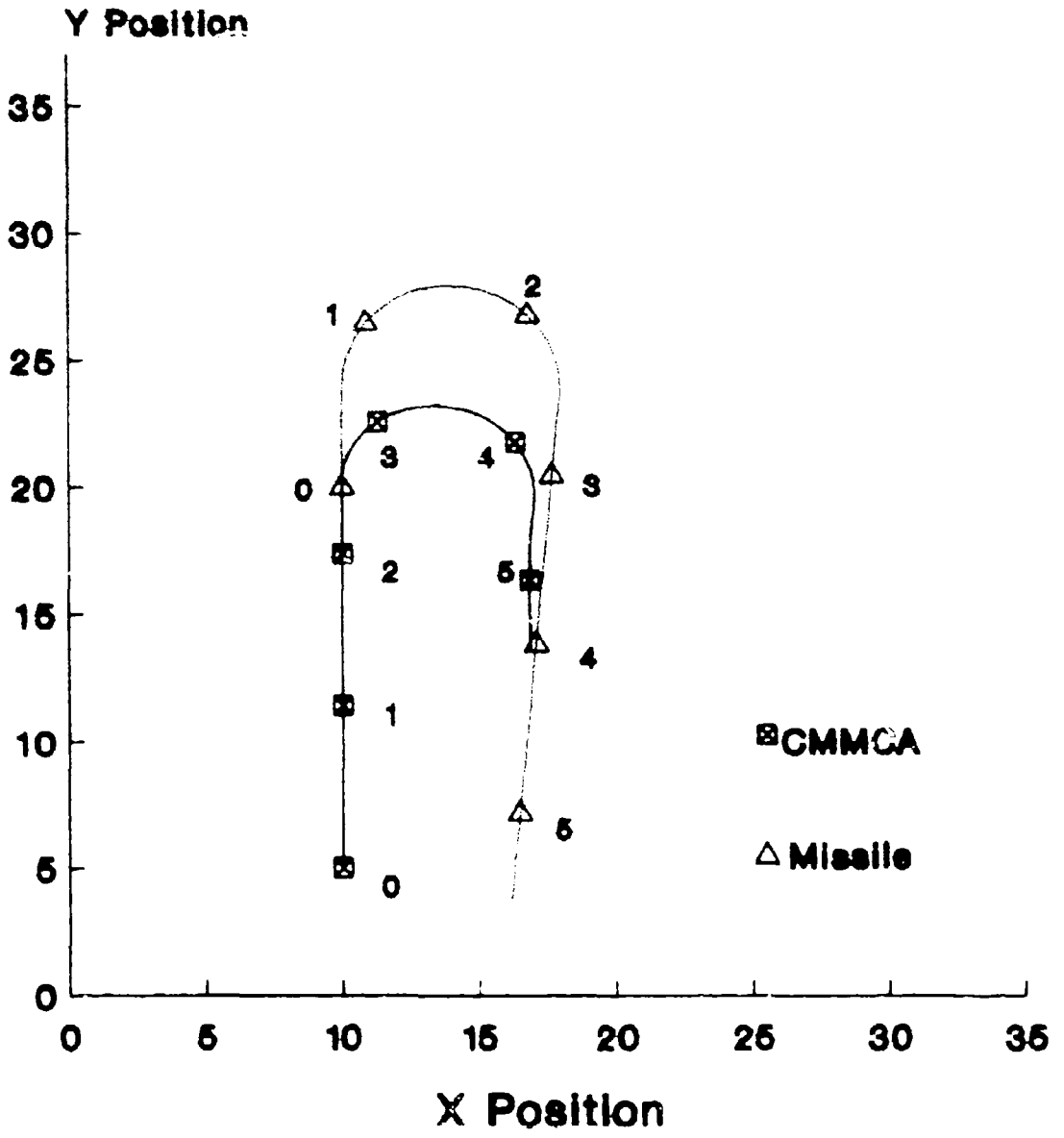

Figure 5. 180 Degree Turn at 0.5 Minute Increments

270 Degree Turn at
0.5 Minute Intervals

	OMCA PARAMETERS				CONTROLS		MISSILE POS		PERFORMANCE	
TNOW	XPOS	YPOS	HDG	VEL	BANK	THRUST	OMX	OMY	RANGE	AZIMUTH
0.00	10.000	5.000	0.	400.	0.0	-1.00	10.000	20.000	15.52	0.00
0.10	10.000	5.664	0.	397.	0.0	-1.00	10.000	20.667	15.53	0.00
0.20	10.000	6.324	0.	394.	0.0	-1.00	10.000	21.333	15.53	0.00
0.30	10.000	6.978	0.	391.	0.0	-1.00	10.000	22.000	15.55	0.00
0.40	10.000	7.628	0.	388.	0.0	-1.00	10.000	22.667	15.56	0.00
0.50	10.000	8.272	0	385.	4.1	-0.86	10.000	23.333	15.58	0.00
0.60	10.008	8.912	1.	382.	4.1	-0.86	10.000	24.000	15.61	-1.26
0.70	10.025	9.547	2.	380.	4.1	-0.86	9.939	24.663	15.64	-2.82
0.80	10.064	10.177	4.	377.	4.1	-0.86	9.768	25.307	15.65	-4.85
0.90	10.112	10.802	5.	375.	4.1	-0.86	9.492	25.913	15.64	-7.34
1.00	10.174	11.421	6.	372.	8.5	-0.16	9.120	26.465	15.60	-10.27
1.10	10.257	12.035	9.	372.	8.5	-0.16	8.660	26.947	15.52	-15.03
1.20	10.368	12.644	12.	371.	8.5	-0.16	8.127	27.346	15.40	-20.25
1.30	10.508	13.246	14.	371.	8.5	-0.16	7.535	27.650	15.24	-25.91
1.40	10.675	13.840	17.	370.	8.5	-0.16	6.900	27.852	15.05	-32.00
1.50	10.870	14.425	20.	370.	-9.2	-0.49	6.241	27.945	14.84	-38.49
1.60	11.059	15.010	17.	368.	-9.2	-0.49	5.575	27.927	14.59	-39.67
1.70	11.218	15.601	14.	367.	-9.2	-0.49	4.922	27.798	14.30	-41.04
1.80	11.346	16.197	11.	365.	-9.2	-0.49	4.299	27.563	13.96	-42.59
1.90	11.443	16.797	8.	364.	-9.2	-0.49	3.724	27.226	13.58	-44.34
2.00	11.509	17.398	5.	362.	-20.8	-0.15	3.213	26.799	13.16	-46.29
2.10	11.519	18.001	358.	362.	-20.8	-0.15	2.781	26.293	12.69	-44.38
2.20	11.457	18.600	351.	361.	-20.8	-0.15	2.439	25.722	12.17	-42.59
2.30	11.322	19.186	344.	361.	-20.8	-0.15	2.196	25.102	11.59	-40.94
2.40	11.117	19.751	337.	360.	-20.8	-0.15	2.060	24.450	10.96	-39.46
2.50	10.845	20.296	330.	360.	-17.3	0.00	2.035	23.784	10.29	-38.20
2.60	10.515	20.767	324.	360.	-17.3	0.00	2.121	23.124	9.59	-38.54
2.70	10.137	21.253	318.	360.	-17.3	0.00	2.318	22.487	8.87	-39.37
2.80	9.714	21.678	313.	360.	-17.3	0.00	2.614	21.892	8.15	-40.86
2.90	9.251	22.059	307.	360.	-17.3	0.00	3.007	21.354	7.45	-43.26
3.00	8.752	22.391	301.	360.	-26.5	-0.10	3.486	20.890	6.78	-46.97
3.10	8.213	22.654	292.	360.	-26.5	-0.10	4.032	20.511	6.17	-48.99
3.20	7.640	22.827	283.	359.	-26.5	-0.10	4.635	20.229	5.64	-53.49
3.30	7.047	22.906	273.	359.	-26.5	-0.10	5.277	20.051	5.22	-61.61
3.40	6.450	22.888	264.	359.	-26.5	-0.10	5.939	19.983	4.97	-74.23
3.50	5.863	22.776	255.	358.	-30.0	0.00	6.606	20.000	4.92	-89.93
3.60	5.306	22.563	244.	358.	-30.0	0.00	7.272	20.024	5.13	-102.01
3.70	4.798	22.250	234.	358.	-30.0	0.00	7.938	20.047	5.54	-108.52
3.80	4.357	21.848	227.	358.	-30.0	0.00	8.604	20.071	6.10	-110.16
3.90	3.998	21.371	212.	358.	-30.0	0.00	9.271	20.094	6.74	-108.56
4.00	3.734	20.836	201.	358.	-30.0	0.00	9.937	20.118	7.42	-104.87

4.10	3.573	20.262	191.	358.	-30.0	0.00	10.603	20.141	8.09	-99.80
4.20	3.522	19.667	180.	358.	-30.0	0.00	11.269	20.165	8.73	-93.76
4.30	3.583	19.073	169.	358.	-30.0	0.00	11.936	20.188	9.33	-86.98
4.40	3.752	18.501	159.	358.	-30.0	0.00	12.602	20.212	9.86	-79.62
4.50	4.025	17.971	148.	358.	-30.0	0.00	13.268	20.235	10.32	-71.75
4.60	4.391	17.500	137.	358.	-30.0	0.00	13.934	20.258	10.71	-63.42
4.70	4.839	17.105	127.	358.	-30.0	0.00	14.601	20.282	11.02	-54.62
4.80	5.352	16.800	116.	358.	-30.0	0.00	15.267	20.305	11.25	-45.37
4.90	5.912	16.596	105.	358.	-30.0	0.00	15.933	20.329	11.42	-35.63
5.00	6.501	16.499	95.	358.	-23.3	-0.18	16.599	20.352	11.52	-25.39
5.10	7.098	16.498	87.	358.	-23.3	-0.18	17.266	20.376	11.59	-17.41
5.20	7.688	16.580	79.	357.	-23.3	-0.18	17.932	20.399	11.64	-9.00
5.30	8.260	16.742	71.	357.	-23.3	-0.18	18.598	20.423	11.68	-0.16
5.40	8.803	16.982	63.	356.	-23.3	-0.18	19.264	20.446	11.72	9.12
5.50	9.307	17.295	55.	356.	13.3	0.04	19.931	20.470	11.79	18.83
5.60	9.804	17.618	59.	356.	13.3	0.04	20.597	20.493	11.86	16.14
5.70	10.325	17.902	63.	356.	13.3	0.04	21.263	20.516	11.94	13.22
5.80	10.866	18.146	68.	356.	13.3	0.04	21.929	20.540	12.01	10.05
5.90	11.424	18.347	72.	356.	13.3	0.04	22.596	20.563	12.07	6.64
6.00	11.996	18.505	77.	356.	9.0	-0.53	23.262	20.587	12.13	3.00
6.10	12.576	18.627	79.	355.	9.0	-0.53	23.928	20.610	12.20	0.61
6.20	13.159	18.718	82.	353.	9.0	-0.53	24.594	20.634	12.27	-1.96
6.30	13.744	18.779	85.	352.	9.0	-0.53	25.261	20.657	12.34	-4.69
6.40	14.328	18.809	88.	350.	9.0	-0.53	25.927	20.681	12.41	-7.38
6.50	14.910	18.808	91.	348.	-0.9	-0.51	26.593	20.704	12.49	-10.63
6.60	15.489	18.796	91.	347.	-0.9	-0.51	27.259	20.728	12.58	-10.42
6.70	16.066	18.786	91.	345.	-0.9	-0.51	27.926	20.751	12.67	-10.19
6.80	16.641	18.780	90.	344.	-0.9	-0.51	28.592	20.774	12.76	-9.94
6.90	17.213	18.777	90.	342.	-0.9	-0.51	29.258	20.798	12.85	-9.67
7.00	17.782	18.777	90.	341.	0.1	-0.31	29.924	20.821	12.95	-9.38
7.10	18.349	18.779	90.	340.	0.1	-0.31	30.591	20.845	13.04	-9.45
7.20	18.915	18.780	90.	339.	0.1	-0.31	31.257	20.868	13.14	-9.51
7.30	19.479	18.781	90.	338.	0.1	-0.31	31.923	20.892	13.24	-9.57
7.40	20.042	18.781	90.	337.	0.1	-0.31	32.589	20.915	13.34	-9.64
7.50	20.603	18.781	90.	336.			33.256	20.939	13.44	-9.70

AVERAGE TIME WAITED DEVIATIONS FROM DESIRED POSITION = 9.120653

270 Degree Turn
0.5 Min Increment

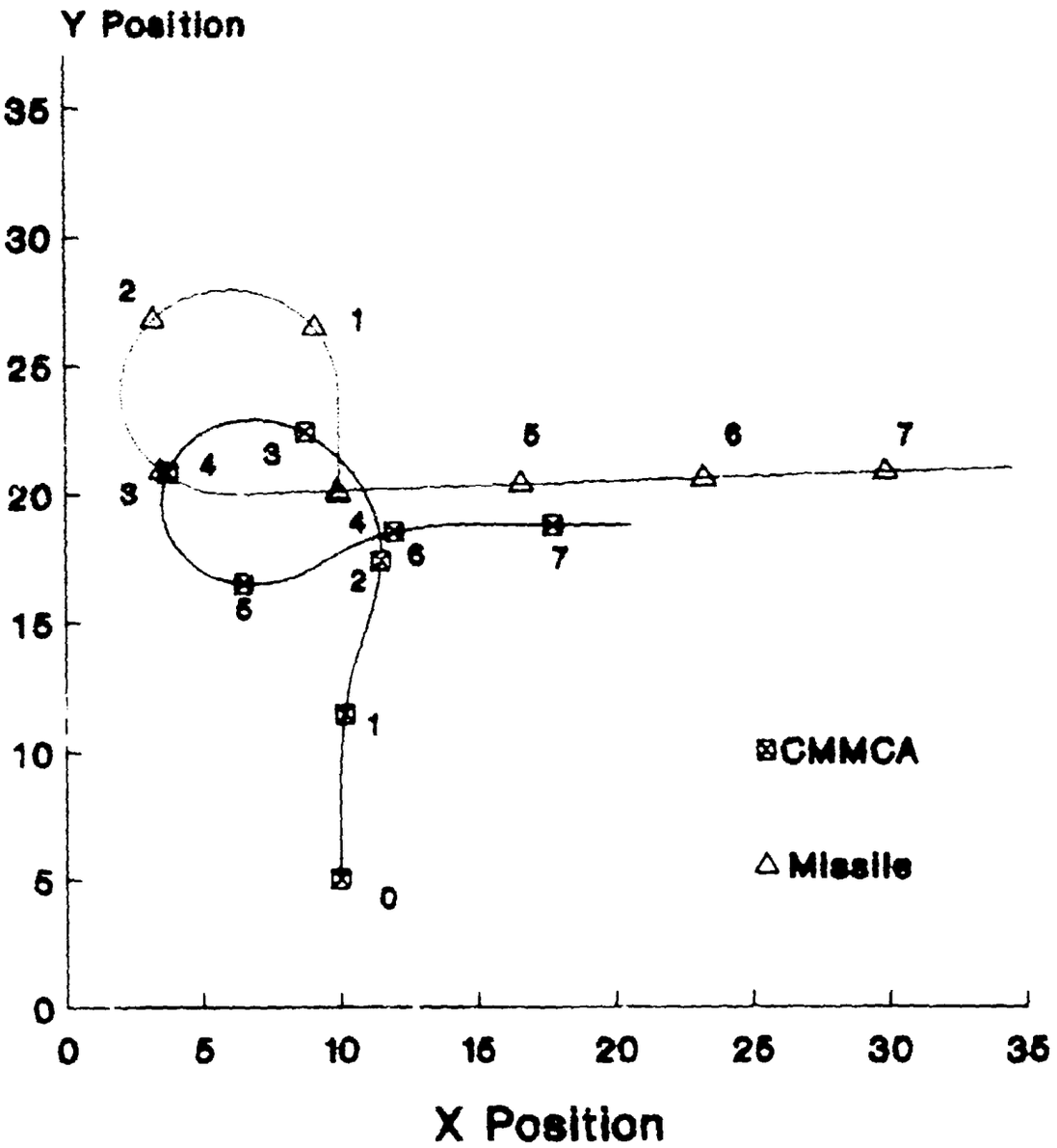

Figure 6. 270 Degree Turn at 0.5 Minute Increments

90 Degree Turn at
0.7 Minute Intervals

	JMMCA PARAMETERS				CONTROLS		MISSILE POS		PERFORMANCE	
TNOW	XPOS	YPOS	HDG	VEL	BANK	THRUST	OMX	OMY	RANGE	AZIMUTH
0.00	10.000	5.000	0.	400.	0.0	-1.00	10.000	20.000	15.52	0.00
0.10	10.000	5.664	0.	397.	0.0	-1.00	10.000	20.667	15.53	0.00
0.20	10.000	6.324	0.	394.	0.0	-1.00	10.000	21.333	15.53	0.00
0.30	10.000	6.978	0.	391.	0.0	-1.00	10.000	22.000	15.55	0.00
0.40	10.000	7.628	0.	388.	0.0	-1.00	10.000	22.667	15.56	0.00
0.50	10.000	8.272	0.	385.	0.0	-1.00	10.000	23.333	15.58	0.00
0.60	10.000	8.911	0.	382.	0.0	-1.00	10.000	24.000	15.61	0.00
0.70	10.000	9.546	0.	379.	0.0	-0.86	10.061	24.663	15.64	0.23
0.80	10.000	10.176	0.	376.	0.0	-0.86	10.232	25.307	15.65	0.88
0.90	10.000	10.801	0.	374.	0.0	-0.86	10.508	25.913	15.64	1.92
1.00	10.000	11.422	0.	371.	0.0	-0.86	10.880	26.465	15.59	3.35
1.10	10.000	12.039	0.	369.	0.0	-0.86	11.340	26.947	15.49	5.13
1.20	10.000	12.651	0.	366.	0.0	-0.86	11.873	27.346	15.34	7.26
1.30	10.000	13.259	0.	363.	0.0	-0.86	12.465	27.650	15.14	9.72
1.40	10.000	13.863	0.	361.	0.0	-0.75	13.100	27.852	14.88	12.49
1.50	10.000	14.463	0.	359.	0.0	-0.75	13.759	27.945	14.56	15.58
1.60	10.000	15.059	0.	356.	0.0	-0.75	14.425	27.927	14.18	18.98
1.70	10.000	15.651	0.	354.	0.0	-0.75	15.089	27.865	13.82	22.62
1.80	10.000	16.240	0.	352.	0.0	-0.75	15.752	27.803	13.52	26.45
1.90	10.000	16.824	0.	350.	0.0	-0.75	16.416	27.741	13.28	30.44
2.00	10.000	17.405	0.	347.	0.0	-0.75	17.080	27.679	13.10	34.57
2.10	10.000	17.982	0.	345.	12.9	0.00	17.744	27.616	12.95	38.79
2.20	10.024	18.557	4.	345.	12.9	0.00	18.407	27.554	12.83	38.57
2.30	10.093	19.128	9.	345.	12.9	0.00	19.071	27.492	12.91	38.21
2.40	10.205	19.692	13.	345.	12.9	0.00	19.735	27.430	12.91	37.70
2.50	10.360	20.245	18.	345.	12.9	0.00	20.399	27.368	12.94	37.02
2.60	10.557	20.785	22.	345.	12.9	0.00	21.062	27.306	13.00	36.14
2.70	10.795	21.309	26.	345.	12.9	0.00	21.726	27.243	13.07	35.06
2.80	11.073	21.812	31.	345.	13.1	0.56	22.390	27.181	13.15	33.77
2.90	11.389	22.294	35.	347.	13.1	0.56	23.054	27.119	13.24	32.19
3.00	11.744	22.751	40.	348.	13.1	0.56	23.717	27.057	13.34	30.42
3.10	12.135	23.182	44.	350.	13.1	0.56	24.381	26.995	13.44	28.46
3.20	12.561	23.583	49.	352.	13.1	0.56	25.045	26.933	13.53	26.32
3.30	13.018	23.952	53.	353.	13.1	0.56	25.709	26.870	13.62	23.98
3.40	13.504	24.287	57.	355.	13.1	0.56	26.372	26.808	13.71	21.47
3.50	14.017	24.584	62.	357.	18.8	-0.29	27.036	26.746	13.79	18.77
3.60	14.556	24.833	68.	356.	18.8	-0.29	27.700	26.684	13.86	13.84
3.70	15.119	25.019	75.	355.	18.8	-0.29	28.364	26.622	13.93	8.60
3.80	15.697	25.142	81.	354.	18.8	-0.29	29.027	26.559	13.99	3.05
3.90	16.283	25.200	87.	353.	18.8	-0.29	29.691	26.497	14.05	-2.79
4.00	16.871	25.192	94.	353.	18.8	-0.29	30.355	26.435	14.12	-8.94

4.10	17.453	25.118	100.	352.	18.8	-0.29	31.019	26.373	14.20	-15.37
4.20	18.022	24.980	107.	351.			31.683	26.311	14.30	-22.08

AVERAGE TIME WEIGHTED DEVIATIONS FROM DESIRED POSITION = 5.292674

90 Degree Turn
0.7 Min Increment

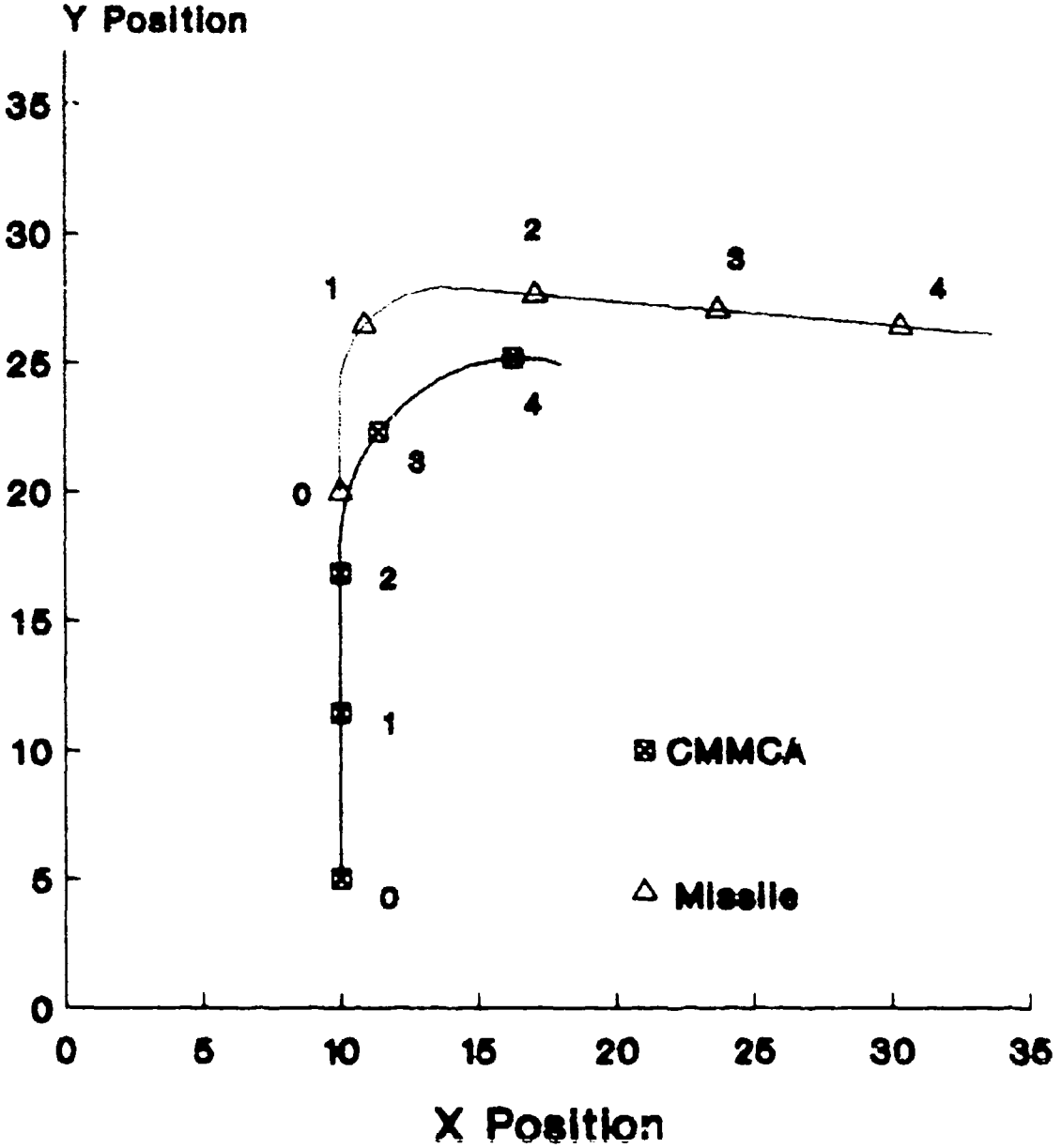

Figure 7. 90 Degree Turn at 0.7 Minute Increments

180 Degree Turn at
0.7 Minute Intervals

	OMEGA PARAMETERS				CONTROLS		MISSILE POS		PERFORMANCE	
TIME	XPOS	YPOS	HDG	VEL	BANK	THRUST	OMX	OMY	RANGE	AZIMUTH
0.00	10.000	5.000	0.	400.	0.0	-1.00	10.000	20.000	15.52	0.00
0.10	10.000	5.664	0.	397.	0.0	-1.00	10.000	20.667	15.53	0.00
0.20	10.000	6.324	0.	394.	0.0	-1.00	10.000	21.333	15.53	0.00
0.30	10.000	6.978	0.	391.	0.0	-1.00	10.000	22.000	15.55	0.00
0.40	10.000	7.628	0.	388.	0.0	-1.00	10.000	22.667	15.56	0.00
0.50	10.000	8.272	0.	385.	0.0	-1.00	10.000	23.333	15.58	0.00
0.60	10.000	8.911	0.	382.	0.0	-1.00	10.000	24.000	15.61	0.00
0.70	10.000	9.546	0.	379.	0.0	-1.00	10.061	24.663	15.64	0.23
0.80	10.000	10.175	0.	376.	0.0	-1.00	10.232	25.307	15.65	0.88
0.90	10.000	10.800	0.	373.	0.0	-1.00	10.508	25.913	15.64	1.92
1.00	10.000	11.419	0.	370.	0.0	-1.00	10.880	26.465	15.59	3.35
1.10	10.000	12.034	0.	367.	0.0	-1.00	11.340	26.947	15.50	5.13
1.20	10.000	12.643	0.	364.	0.0	-1.00	11.873	27.346	15.35	7.26
1.30	10.000	13.247	0.	361.	0.0	-1.00	12.465	27.650	15.15	9.71
1.40	10.009	13.847	0.	358.	0.0	-0.80	13.100	27.852	14.89	12.48
1.50	10.000	14.442	0.	356.	0.0	-0.80	13.759	27.945	14.58	15.56
1.60	10.000	15.033	0.	353.	0.0	-0.80	14.425	27.927	14.21	18.94
1.70	10.000	15.619	0.	351.	0.0	-0.80	15.078	27.798	13.79	22.63
1.80	10.000	16.202	0.	348.	0.0	-0.80	15.701	27.563	13.32	26.65
1.90	10.000	16.781	0.	346.	0.0	-0.80	16.276	27.226	12.83	31.00
2.00	10.000	17.356	0.	344.	0.0	-0.80	16.787	26.799	12.30	35.70
2.10	10.000	17.927	0.	341.	9.2	-0.01	17.219	26.293	11.75	40.79
2.20	10.017	18.495	3.	341.	9.2	-0.01	17.561	25.722	11.19	43.06
2.30	10.066	19.062	6.	341.	9.2	-0.01	17.804	25.102	10.60	45.68
2.40	10.146	19.624	10.	341.	9.2	-0.01	17.940	24.450	10.00	48.73
2.50	10.257	20.182	13.	341.	9.2	-0.01	17.965	23.784	9.40	52.27
2.60	10.398	20.732	16.	341.	9.2	-0.01	17.909	23.120	8.84	56.51
2.70	10.570	21.274	19.	341.	9.2	-0.01	17.849	22.456	8.39	61.75
2.80	10.772	21.805	22.	341.	30.0	0.00	17.789	21.792	8.08	67.91
2.90	11.042	22.305	33.	341.	30.0	0.00	17.729	21.128	7.88	66.53
3.00	11.403	22.742	45.	341.	30.0	0.00	17.669	20.464	7.77	65.29
3.10	11.844	23.100	56.	341.	30.0	0.00	17.609	19.800	7.75	63.85
3.20	12.345	23.365	67.	341.	30.0	0.00	17.549	19.136	7.81	61.92
3.30	12.889	23.527	78.	341.	30.0	0.00	17.489	18.472	7.92	59.27
3.40	13.454	23.581	90.	341.	30.0	0.00	17.429	17.808	8.07	55.77
3.50	14.018	23.523	101.	341.	24.5	0.17	17.369	17.144	8.24	51.37
3.60	14.565	23.368	110.	341.	24.5	0.17	17.309	16.480	8.42	48.48
3.70	15.082	23.130	119.	342.	24.5	0.17	17.249	15.816	8.61	44.84
3.80	15.557	22.816	128.	342.	24.5	0.17	17.189	15.153	8.80	40.48
3.90	15.979	22.431	136.	343.	24.5	0.17	17.129	14.489	8.97	35.42
4.00	16.337	21.986	145.	343.	24.5	0.17	17.069	13.825	9.12	29.71

4.10	16.623	21.490	154.	344.	24.5	0.17	17.009	13.161	9.25	23.38
4.20	16.830	20.956	163.	344.	11.5	-0.25	16.949	12.497	9.36	16.43
4.30	16.979	20.402	167.	344.	11.5	-0.25	16.889	11.833	9.46	13.91
4.40	17.090	19.841	171.	343.	11.5	-0.25	16.829	11.169	9.55	11.10
4.50	17.161	19.275	175.	342.	11.5	-0.25	16.769	10.505	9.65	8.00
4.60	17.194	18.706	179.	341.	11.5	-0.25	16.709	9.841	9.74	4.62
4.70	17.187	18.138	182.	341.	11.5	-0.25	16.649	9.177	9.83	0.97
4.80	17.141	17.573	186.	340.	11.5	-0.25	16.589	8.513	9.92	-2.95
4.90	17.056	17.013	190.	339.	-6.9	-0.20	16.529	7.849	10.01	-7.12
5.00	16.967	16.456	188.	338.	-6.9	-0.20	16.469	7.185	10.11	-4.95
5.10	16.901	15.896	186.	338.	-6.9	-0.20	16.409	6.521	10.20	-2.63
5.20	16.859	15.335	183.	337.	-6.9	-0.20	16.349	5.857	10.30	-0.16
5.30	16.840	14.774	181.	337.	-6.9	-0.20	16.289	5.193	10.40	2.45
5.40	16.845	14.213	178.	336.	-6.9	-0.20	16.229	4.529	10.50	5.20
5.50	16.873	13.654	176.	335.	-6.9	-0.20	16.169	3.865	10.60	8.09
5.60	16.924	13.098	174.	335.			16.109	3.201	10.71	11.10

AVERAGE TIME WEIGHTED DEVIATIONS FROM DESIRED POSITION = 7.939556

180 Degree Turn
0.7 Min Increment

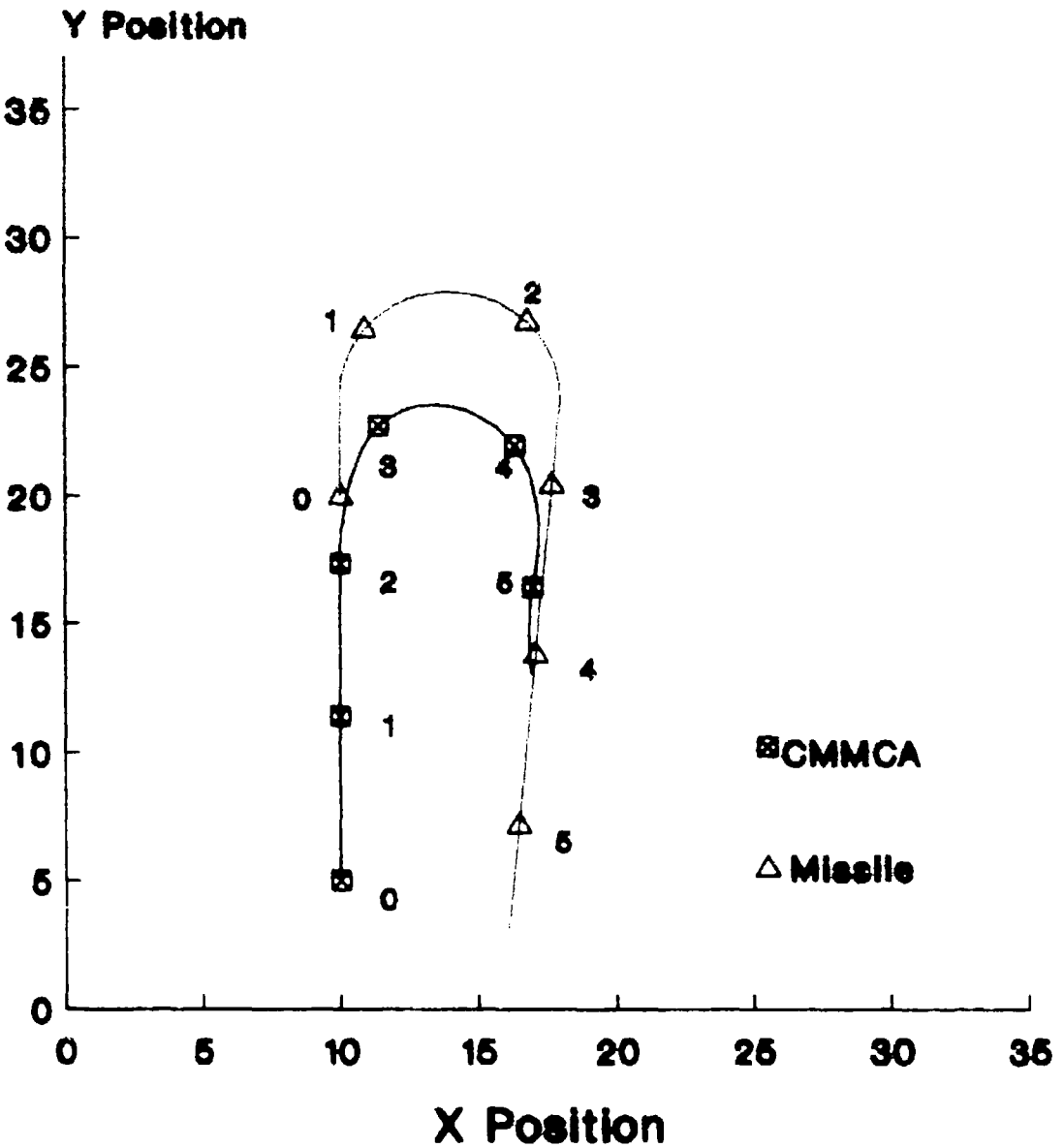

Figure 8. 180 Degree Turn at 0.7 Minute Increments

270 Degree Turn at
0.7 Minute Intervals

	OMICA PARAMETERS				CONTROLS		MISSILE POS		PERFORMANCE	
TIME	XPOS	YPOS	HDG	VEL	BANK	THRUST	OMX	OMY	RANGE	AZIMUTH
0.00	10.000	5.000	0.	400.	0.0	-1.00	10.000	20.000	15.52	0.00
0.10	10.000	5.664	0.	397.	0.0	-1.00	10.000	20.667	15.53	0.00
0.20	10.000	6.324	0.	394.	0.0	-1.00	10.000	21.333	15.53	0.00
0.30	10.000	6.978	0.	391.	0.0	-1.00	10.000	22.000	15.55	0.00
0.40	10.000	7.628	0.	388.	0.0	-1.00	10.000	22.667	15.56	0.00
0.50	10.000	8.272	0.	385.	0.0	-1.00	10.000	23.333	15.58	0.00
0.60	10.000	8.911	0.	382.	0.0	-1.00	10.000	24.000	15.61	0.00
0.70	10.000	9.546	0.	379.	-2.7	-0.91	9.939	24.663	15.64	-0.23
0.80	9.995	10.176	359.	376.	-2.7	-0.91	9.768	25.307	15.65	-0.02
0.90	9.981	10.800	358.	374.	-2.7	-0.91	9.492	25.913	15.64	-0.17
1.00	9.958	11.421	357.	371.	-2.7	-0.91	9.120	26.465	15.59	-0.66
1.10	9.925	12.036	357.	368.	-2.7	-0.91	8.660	26.947	15.49	-1.47
1.20	9.884	12.646	356.	365.	-2.7	-0.91	8.127	27.346	15.34	-2.57
1.30	9.834	13.251	355.	363.	-2.7	-0.91	7.535	27.650	15.12	-3.96
1.40	9.776	13.850	354.	360.	-22.6	-0.20	6.900	27.852	14.84	-5.62
1.50	9.669	14.440	346.	359.	-22.6	-0.20	6.261	27.945	14.50	-0.56
1.60	9.485	15.009	339.	359.	-22.6	-0.20	5.575	27.927	14.08	4.54
1.70	9.227	15.547	331.	358.	-22.6	-0.20	4.922	27.798	13.59	9.74
1.80	8.899	16.045	323.	358.	-22.6	-0.20	4.299	27.563	13.03	15.06
1.90	8.508	16.493	315.	357.	-22.6	-0.20	3.724	27.226	12.41	20.55
2.00	8.061	16.884	308.	356.	-22.6	-0.20	3.213	26.799	11.74	26.28
2.10	7.566	17.211	300.	356.	9.9	0.00	2.781	26.293	11.02	32.32
2.20	7.061	17.522	303.	356.	9.9	0.00	2.439	25.722	10.23	27.44
2.30	6.575	17.862	306.	356.	9.9	0.00	2.196	25.102	9.36	22.44
2.40	6.109	18.228	310.	356.	9.9	0.00	2.060	24.430	8.43	17.30
2.50	5.665	18.620	313.	356.	9.9	0.00	2.035	23.784	7.47	12.01
2.60	5.243	19.037	316.	356.	9.9	0.00	2.121	23.124	6.52	6.49
2.70	4.846	19.477	319.	356.	9.9	0.00	2.316	22.487	5.61	0.57
2.80	4.474	19.939	323.	356.	30.0	0.00	2.614	21.892	4.82	-6.24
2.90	4.165	20.444	333.	356.	30.0	0.00	3.007	21.354	4.26	-25.23
3.00	3.956	20.998	344.	356.	30.0	0.00	3.484	20.880	4.03	-87.13
3.10	3.854	21.582	355.	356.	30.0	0.00	4.032	20.511	4.14	175.64
3.20	3.862	22.174	6.	356.	30.0	0.00	4.635	20.229	4.51	152.66
3.30	3.981	22.754	16.	356.	30.0	0.00	5.277	20.051	5.00	137.96
3.40	4.206	23.302	27.	356.	30.0	0.00	5.939	19.983	5.48	125.24
3.50	4.530	23.798	38.	356.	29.5	-0.02	6.606	20.000	5.89	113.38
3.60	4.940	24.225	49.	356.	29.5	-0.02	7.272	20.024	6.25	102.45
3.70	5.421	24.571	59.	356.	29.5	-0.02	7.938	20.047	6.54	91.82
3.80	5.957	24.822	70.	356.	29.5	-0.02	8.604	20.071	6.75	81.22
3.90	6.530	24.971	80.	356.	29.5	-0.02	9.271	20.094	6.88	70.44
4.00	7.120	25.012	91.	356.	29.5	-0.02	9.937	20.118	6.92	59.29

4.10	7.708	24.944	101.	355.	29.5	-0.02	10.603	20.141	6.89	47.56
4.20	8.273	24.769	112.	355.	16.2	-0.26	11.269	20.165	6.80	35.02
4.30	8.810	24.520	117.	355.	16.2	-0.26	11.936	20.188	6.67	26.84
4.40	9.319	24.222	123.	354.	16.2	-0.26	12.602	20.212	6.55	17.93
4.50	9.797	23.878	128.	353.	16.2	-0.26	13.268	20.235	6.43	8.17
4.60	10.239	23.490	134.	352.	16.2	-0.26	13.934	20.258	6.33	-2.49
4.70	10.642	23.064	139.	351.	16.2	-0.26	14.601	20.282	6.28	-14.04
4.80	11.001	22.602	145.	351.	16.2	-0.26	15.267	20.305	6.28	-26.32
4.90	11.313	22.109	150.	350.	-21.9	-0.23	15.933	20.329	6.37	-39.03
5.00	11.639	21.627	142.	349.	-21.9	-0.23	16.599	20.352	6.50	-38.05
5.10	12.026	21.193	135.	348.	-21.9	-0.23	17.266	20.376	6.64	-35.95
5.20	12.466	20.816	127.	348.	-21.9	-0.23	17.932	20.399	6.79	-32.79
5.30	12.952	20.501	119.	347.	-21.9	-0.23	18.598	20.423	6.92	-28.68
5.40	13.475	20.255	112.	346.	-21.9	-0.23	19.264	20.446	7.04	-23.67
5.50	14.024	20.081	104.	346.	-21.9	-0.23	19.931	20.470	7.14	-17.83
5.60	14.591	19.983	96.	345.	-3.7	-0.39	20.597	20.493	7.23	-11.19
5.70	15.162	19.927	95.	344.	-3.7	-0.39	21.263	20.516	7.32	-10.62
5.80	15.733	19.883	94.	343.	-3.7	-0.39	21.929	20.540	7.40	-9.90
5.90	16.302	19.851	93.	342.	-3.7	-0.39	22.596	20.563	7.49	-9.05
6.00	16.870	19.832	91.	340.	-3.7	-0.39	23.262	20.587	7.58	-8.07
6.10	17.437	19.826	90.	339.	-3.7	-0.39	23.928	20.610	7.67	-6.96
6.20	18.001	19.832	89.	338.	-3.7	-0.39	24.594	20.634	7.75	-5.73
6.30	18.563	19.851	88.	337.	0.0	-0.13	25.261	20.657	7.84	-4.40
6.40	19.124	19.875	88.	336.	0.0	-0.13	25.927	20.681	7.93	-4.27
6.50	19.684	19.899	87.	336.	0.0	-0.13	26.593	20.704	8.02	-4.14
6.60	20.243	19.924	87.	336.	0.0	-0.13	27.259	20.728	8.12	-4.02
6.70	20.802	19.949	87.	335.	0.0	-0.13	27.926	20.751	8.21	-3.89
6.80	21.360	19.973	87.	335.	0.0	-0.13	28.592	20.774	8.30	-3.77
6.90	21.917	19.998	87.	335.	0.0	-0.13	29.258	20.798	8.40	-3.65
7.00	22.474	20.023	87.	334.	1.0	-0.08	29.924	20.821	8.49	-3.53
7.10	23.030	20.047	88.	334.	1.0	-0.08	30.591	20.845	8.59	-3.78
7.20	23.586	20.067	88.	334.	1.0	-0.08	31.257	20.868	8.69	-4.05
7.30	24.142	20.084	88.	333.	1.0	-0.08	31.923	20.892	8.79	-4.34
7.40	24.697	20.097	89.	333.	1.0	-0.08	32.589	20.915	8.89	-4.66
7.50	25.252	20.108	89.	333.	1.0	-0.08	33.256	20.939	8.99	-5.00
7.60	25.806	20.115	89.	333.	1.0	-0.08	33.922	20.962	9.09	-5.37
7.70	26.350	20.119	90.	332.			34.588	20.986	9.19	-5.75

AVERAGE TIME WEIGHTED DEVIATIONS FROM DESIRED POSITION = 8.872969

270 Degree Turn
0.7 Min Increment

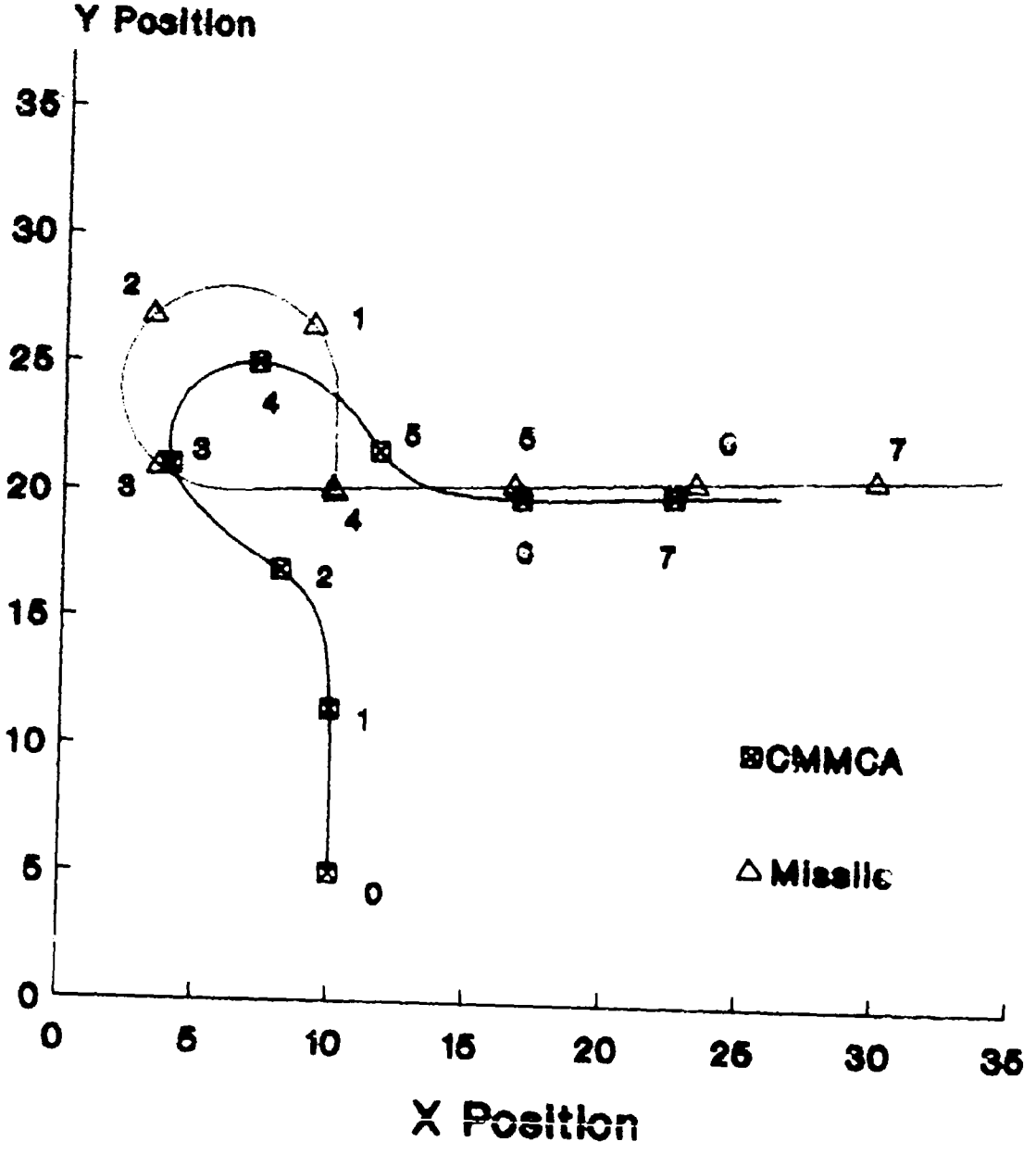

Figure 9. 270 Degree Turn at 0.7 Minute Increments

Bibliography

1. Bellman, Richard E. Dynamic Programming. Princeton, New Jersey: Princeton University Press, 1957.

2. Bellman, Richard E. and Stuart E. Dreyfus. Applied Dynamic Programming. Princeton, New Jersey: Princeton University Press, 1962.

3. Dommasch, Daniel and others. Airplane Dynamics (Third Edition). New York: Pitman Publishing corporation, 1961.

4. Etkin, Bernard. Dynamics of Atmospheric Flight. New York: John Wiley & Sons, Inc., 1972.

5. Horton, Capt Robert W. Cruise Missile Project Manager. Personal Interview. 4950th Test Wing, Wright-Patterson AFB, OH, 10 November 1988.

6. Larson, Robert E. "Dynamic Programming with Reduced Computational Requirements," IEEE Transactions on Automatic Control, AC-10: 135-143 (April 1965).

7. Larson, Robert E. State Increment Dynamic Programming. New York: American Elsevier Publishing Company, Inc., 1968.

8. Larson, Robert E. and John L. Casti. Principles of Dynamic Programming: Part II. New York: Marcel Dekker, Inc., 1982.

9. Microsoft Corporation. Microsoft FORTRAN Compiler for the MS-DOS Operating System: User's Guide. St. Joseph, Michigan: Zenith Data Systems Corporation, 1984.

10. Schuppe, Thomas F. Analysis of CMMCA Tracking System. Unpublished report. Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, 1988.

VITA

Captain Leonard G. Heavner was born on 20 February 1960 in Limestone, Maine. He graduated from high school in Sacramento, California, in 1977 and atteded the United States Air Force Academy, from which he received the degree of Bachelor of Science in Computer Science in May 1981. After graduation, he attended Undergraduate Pilot Training at Columbus, Mississippi, and received his wings in June 1982. He then served as a C-141 pilot and flight instructor in the 41st Military Airlift Squadron, Charleston AFB, SC, until entering the School of Engineering, Air Force Institute of Technology, in August 1987.

Permanent address: 305 Homeland Dr
Martinsburg, WV 25401

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GST/ENS/89M-6	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (If applicable) AFIT/ENS	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Air Force Institute of Technology (AU) Wright-Patterson AFB, Ohio 45433-6583	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.
11. TITLE (Include Security Classification) See Box 19	
12. PERSONAL AUTHOR(S) Leonard G. Heavner, B.S., Capt, USAF	
13a. TYPE OF REPORT MS Thesis 13b. TIME COVERED FROM _____ TO _____ 14. DATE OF REPORT (Year, Month, Day) 1989 March 15. PAGE COUNT 115	
16. SUPPLEMENTARY NOTATION	

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	
12	04		Dynamic Programming, Tracking, Cruise Missiles, Optimization

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Title: AN INVESTIGATION INTO THE APPLICATION OF DYNAMIC PROGRAMMING TO DETERMINE OPTIMAL CONTROLS FOR TRACKING A CRUISE MISSILE MANUEVER BY CMMCA

Thesis Chairman: Thomas F. Schuppe, Lt Col, USAF
Associate Professor of Operations Research

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL THOMAS P. SCHUPPE, Lt Col, USAF	22b. TELEPHONE (Include Area Code) (513) 255-5758 22c. OFFICE SYMBOL AFIT/ENS

This study investigated the application of dynamic programming to determine the optimal control inputs for tracking a cruise missile manuever by the Cruise Missile Mission Control Aircraft (CMMCA). Conventional dynamic programming and state increment dynamic programming were considered. The objective was to minimize cruise missile deviations from a desired position within the radar sweep cone. The effect of the time increment size defining the stages of the methodology was evaluated.

The dynamic programming model formulated proved to be extremely sensitive to the time increment used. Widely different results were obtained by the increments investigated. The research concluded that a better definition of an objective function for the CMMCA tracking problem is required.